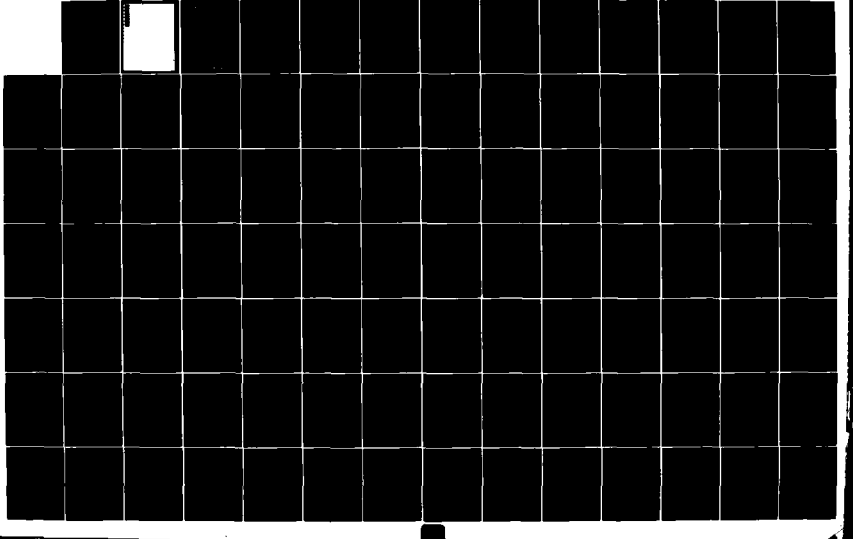
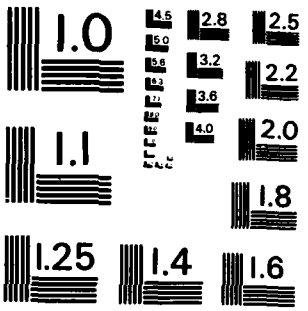
AD-A121 018 ANALYTIC PERFORMANCE MODELING OF IMAGING TRACKERS(U) 1/2
BATTELLE COLUMBUS LABS OH R L FREY ET AL. 08 MAR 82
MAC-FR82-72-784 DAAG29-76-D-0100

UNCLASSIFIED F/G 20/6 NL





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A121015	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* ANALYTIC PERFORMANCE MODELING OF IMAGING TRACKERS		5. TYPE OF REPORT & PERIOD COVERED FINAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. L. FREY, C. D. NEALY; L. M. RUBIN, R. M. WILCOX		8. CONTRACT OR GRANT NUMBER(s) DAAG29-76-0100
9. PERFORMING ORGANIZATION NAME AND ADDRESS IMAGE PROCESSING LAB., E-O ENGINEERING DIVISION HUGHES AIRCRAFT COMPANY EL SEGUNDO, CALIFORNIA 90245		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS BATTELLE COLUMBUS LABORATORY 505 KING AVENUE COLUMBUS, OHIO 43201		12. REPORT DATE MARCH 1982
		13. NUMBER OF PAGES 164
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* U.S. ARMY NIGHT VISION AND ELECTRO-OPTICS LAB. FORT BELVOIR, VIRGINIA 22060		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT *(of this Report)*		
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)* IMAGE PROCESSING, IMAGING TRACKERS, KALMAN FILTERING, NON-LINEAR FILTERING, MACSYMA		
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)* ANALYTIC METHODS HAVE BEEN USED TO BOUND THE PERFORMANCE OF THE OPTIMAL IMAGING TRACKER.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ANALYTIC PERFORMANCE MODELING OF IMAGING TRACKERS

MARCH 5, 1982

Delivery Order No. 1943

R. L. FREY

C. D. NEALY

L. M. RUBIN

R. M. WILCOX

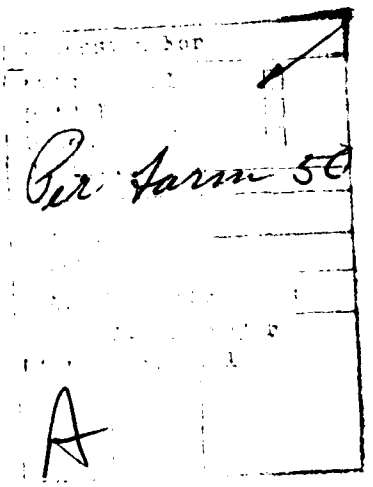


TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1. INTRODUCTION

This report will describe progress made on the STAS Contract with respect to providing analytical bounds and approximations to the performance modeling of electro-optical target trackers.

The major goal of this sutdy was to analyze those limits that one can place on an imaging tracker performance in terms of fundamental properties of the sensor and the target such as vehicular dynamics and uncertainty, signal to noise of the sensor, and resolution. To accomplish these goals it was necessary to develop a complete imaging tracker system model including a model for the relevant scene, sensor, and observation, as well as a model for the tracker processor. It was also necessary to establish those measures and criteria by which a tracker's performance can be gauged so that one can produce from these analyses a systematic design procedure applicable to the design, development, and specification of tracker systems. A number of analytic tools that were seen as potentially applicable to tracker analysis were evaluated and a subset of these that were thought to have particular application were chosen. These tools were then applied to the models to produce bounds and approximations to the actual performance one could expect of a tracker in a given environment.

The tasks that were performed under this contract include:

1. Providing a simple analytical scene model, including the effect of target maneuvering, target shape variation with position, and clutter obscuration, as well as sensor degradations due to e.g., sensor blur, noise and sampling.

2. Developing a model that is capable of bounding the optimal performance of a tracker. The way this was arrived at was

to define in formal mathematical terms the function of an optimal tracker and then producing the equations governing this function to which there is a uniquely defined exact solution. This solution to the optimal tracker equation was then approximated and bounded by looking at properties of this solution and relating it to the performance parameters of the tracker.

3. Establishing a set of parameters which were capable of gauging the tracker performance. As expected, these tracker performance parameters varied with the types of missions and scenarios that one would require the use of a tracker. A number of these performance parameters have been defined, and these parameters have been applied to sample tracker system sensors and the use of the measures have been described.

4. Taking all of the key results of the preceding three tasks and verifying them by direct computer simulation. This was performed by taking the equation referred to in (2) and numerically solving them and then comparing the resultant solutions to our derived set of approximations and bounds.

5. Exploring the viability of these techniques in actual tracker applications. There is included in this report an example of how these techniques can be used for approximating, for example, aimpoint jitter for a given precision tracking problem.

Figure 1-1 shows the block diagram that will be employed to model tracker systems from a general standpoint. It starts off with the scene, which can contain a number of targets and clutter objects as well as various other man-made non-clutter objects. The scene is then modeled as having targets having a given set of dynamics and a given thermal signature. The effect of the thermal signature, the projection of the scene, and the emitted radiation from the relevant objects

in the scene forms with the sensor model of such effects as blur and noise, the sensed image produced by the sensor. This image is then fed into a tracker processor. The processor is modeled as implicitly providing a state probability density. This may not be an actual output of the tracker, but it is treated in our model as if this density were operated on to provide a target state estimate known as $\hat{X}$. This target state estimate can then be used to drive the servomechanism which can control a set of gimbals, which can then be fed back through a gimbal pickoff to feed a set of derived angles into the tracker. There are two types of trackers one can describe, known as either open loop or closed loop. In the closed loop case, the servomechanism directly drives the gimbals which controls the pointing of the sensor and which can feed information to the tracker as to the new sensor line of sight position. The open loop case is tracking "in raster" and basically treating the sensor as if it were stationary. It can be seen that the transformation of an open loop to a closed loop tracker invloves including a certain amount of pickoff noise and analyzing the relevant servomechanisms. Although these problems are non-trivial, they are beyond the scope of this particular report since it is felt that the analysis of servomechanisms is treated in sufficient detail in many other publications and reports. The open loop case will therefore be treated in this report although the generalization of our formulation to the closed loop case will be apparent.

To discuss in more detail the modeling of a tracker, refer to Figure 1-2, where a series of sensed images is represented as the vector function $\vec{y}(t)$ and the tracker, as previously discussed, produces a probability estimate of the state $\vec{x}(t)$, $P(\vec{x}(t),t)$. $\vec{x}(t)$ represents the state of the targets and of the scene that one is tracking. The state includes such things as target position, velocity, and acceleration, in addition to clutter positions and obscuration co-efficients. In general, the state vector changes as a function of time. The resultant state probability density can then be operated on by an arbitrary functional operator producing a state estimate. Examples of the types of estimators one can use are, e.g., a minimum mean square estimate which would take the mean of the underlying probability distributions, or a maximum a priori estimator which could pick the peak of the probability density. One could also define a minimum error probability estimator or a mini-max type function as well in order to produce an estimate that minimizes the criteria of particular relevance in the scenario of interest.

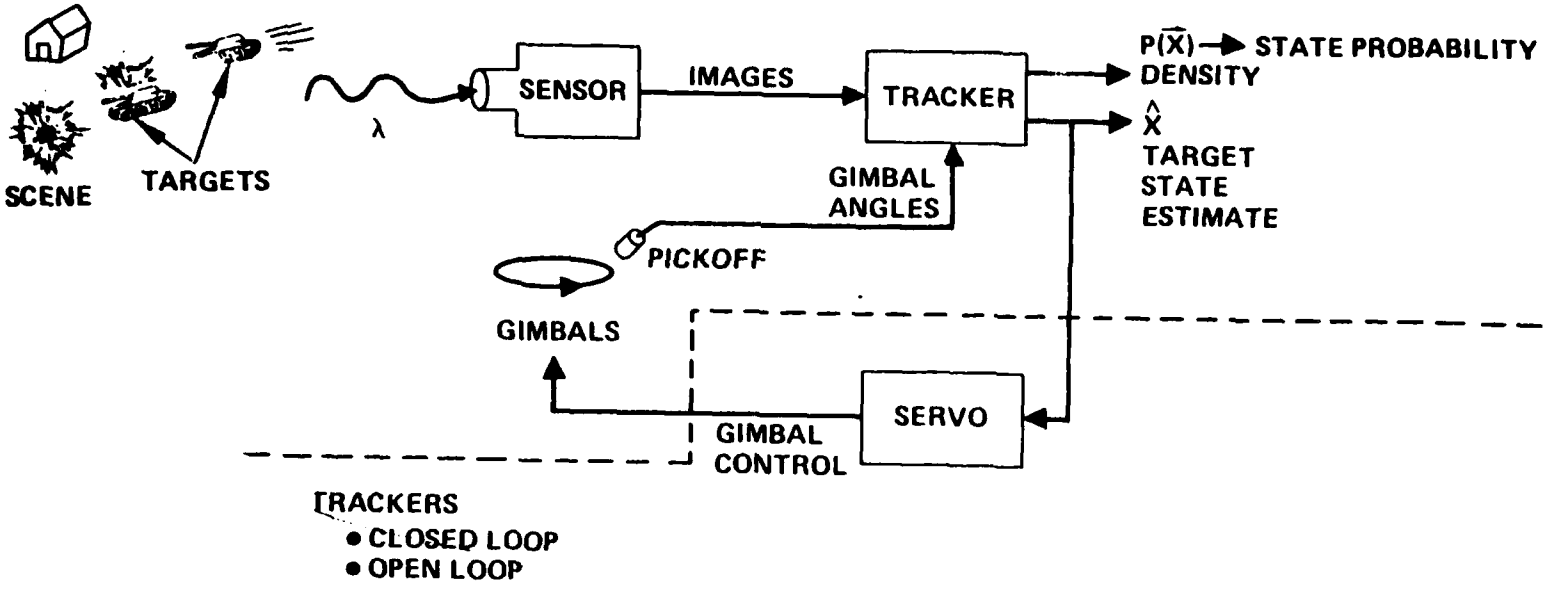
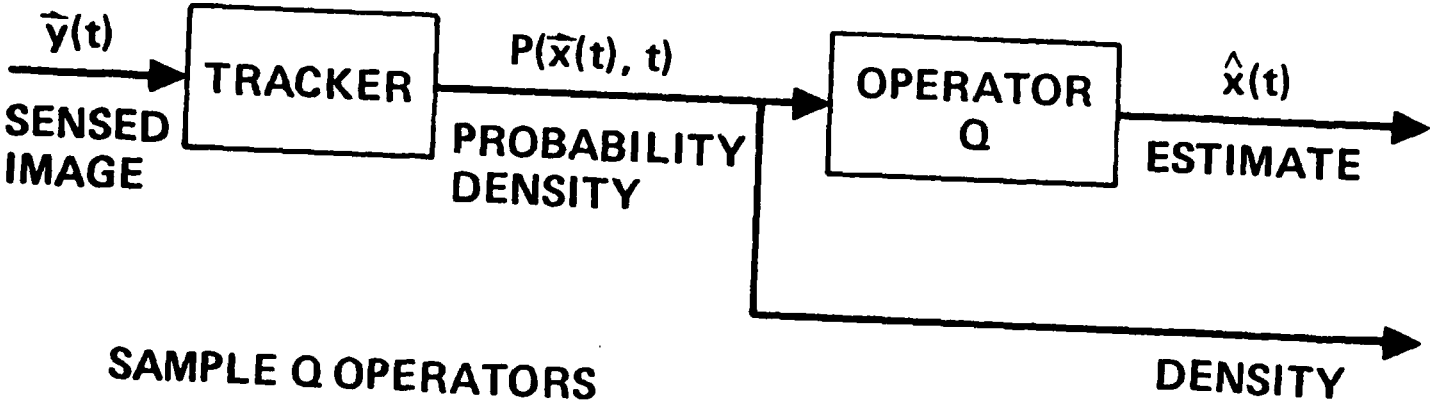


FIGURE 1-1. TRACKER BLOCK DIAGRAM



MMSE $Q(P) = \int \tilde{x} P(x(t), t) d\tilde{x}$

MAP $Q(P) = \hat{x}$ s.t. $P(\hat{x}, t) \geq P(U, t)\ \forall \tilde{U}$

FIGURE 1-2. TRACKER MODELING

This report will be broken into eight sections. After the introduction, the second section describes the analytic techniques that have potential value for the modeling of tracker systems in various environments. These techniques include the ways of analyzing the stochastic differential equations that are central to the definition of the tracker problem. The third section will describe the target model including target dynamics, maneuvering, and shape. The fourth section is related to scene modeling and will describe the techniques for taking the scene state, (which includes the target state as well as the clutter and obscuration states) and producing from the target, clutter and scensor models, an expression for what the sensed IR Scene will be in terms of the sampled output. The fifth section deals with bounds and approximations and will describe various techniques for bounding and approximating the performance of the optimal filter operating on the video provided by the scene model. Following this, the sixth section, titled Simulation, will describe the computer program that has been developed to directly simulate the exact solution to arbitrary non-linear stochastic differential equation, how it has been applied to the problems of interest, and some of the results that have been obtained. In the seventh section, system applications of the techniques described in this report will be discussed. And finally, a summary and conclusion of the contents of the report will be given.

SECTION 2. BASIC ANALYTIC TECHNIQUES

In this section, the basic analytic techniques that can be used to model, bound and approximate the performance of FLIR trackers are summarized. First, the fundamental nature of the tracker modeling problem is discussed, along with a definition of the interpretation of the meaning of the word 'tracker' used in this report. A set of representative tracker evaluation parameters and normalizing system parameters are then given. This leads to a formulation of the problem in terms of stochastic differential equations. The basic characteristics of these types of equations are summarized along with the two types of disturbance processes, Wiener and Poisson, applicable to the tracker problem. Then the extent to which a solution can be found is discussed in terms of Kushner's equation and its reduction to the Kalman filter in the linear case. Finally, approaches for computing performance bounds and approximations for certain classes of optimal nonlinear trackers are presented and discussed.

2.1 TRACKER MODELING PROBLEM

Before the fundamental nature of the tracker modeling problem can be defined, it is necessary to define the meaning of the word 'tracker', in the context of this analysis study. In general terms, trackers can be classified as either open-loop or closed-loop. With closed loop trackers, the sensor is driven to maintain the target in a fixed position in the sensor field of view. In an open-loop tracker, the location of the target in the FOV is tracked as it moves through the FOV. In this report, the results will be derived for an open-loop type tracker operating on FLIR video of a fixed, noisy, cluttered scene with a target moving through the scene. This restriction is made for simplicity of development, with no real loss of generality since the results and techniques developed can easily be extended.

The basic tracker modeling problem is to develop a mathematical model of tracker performance that allows extraction of figures of merit for use in comparing the relative performance of various tracker implementations with bounds on optimal tracker performance for various applications. The model must be broad enough to encompass the analysis of relatively simple centroid trackers, correlation trackers, and advanced, intelligent target trackers now under development (cf ref (27)). The latter class of trackers use modern image processing and pattern recognition techniques to initially acquire targets and to predict or mitigate occlusion effects of clutter also present in the scene. In this sense, this problem is a natural extension of a previous study on autocuer modeling analysis (ref (30)). The extension is basically to a time history of observations being used to predict future time history.

In order to illustrate the scope of application and its affect on appropriate figures of merit, two examples of the use of FLIR trackers are pertinent: High Energy Lasers (HEL), and advanced tank fire control.

In the HEL application, the requirement is to provide automatic aimpoint selection and maintenance for the HEL beam. Performing this function requires, in turn, highly accurate position tracking to maintain the beam on one spot; and target shape and orientation tracking to find the proper vulnerable aimpoint. Accurate target dynamics prediction is important to the extent that it affects the tracker stability.

In the advanced tank fire control application, the requirements will be to automatically find, identify, and track targets; while providing capability to hit maneuvering targets and automatically position the gun to hit the target. Here the critical requirement is to accurately predict the target location in the future at the intercept point, typically one to two seconds. Meeting this requirement requires accurate modeling and estimation of target dynamics. The effect of not properly modeling target dynamics is illustrated in (32), where it is shown that an extended Kalman filter implementation of a tracker could not follow an abruptly maneuvering target in the air-air scenario.

It is apparent from these two examples that different figures of merit apply to different applications. The appropriate list of figures of merit, however, can be easily identified:

1) Expected RMS tracker error
2) Stability (probability of losing lock)
3) Prediction accuracy
4) Robustness (model sensitivity)

These figures of merit will be functions of the conditions under which the tracker will operate:

1) Signal to noise ratio

2) Clutter density

3) Target maneuver parameters

4) Target size

5) Degree of non-linearity

In order to relate the figures of merit to these parameters, a model of the tracker performance must be developed. This can be done formally as follows:

In section 3, a 3-D target dynamics model is developed which has the form

$$d\vec{x} = f(\vec{x}, t)dt + d\vec{\omega} \tag{2-1}$$

where $\vec{x}$ is the target dynamics state vector (augmented by clutter states as described in section 4) and $\vec{w}$ is an appropriate random process. Also in section 4, a 2-dimensional projection of a target at state $\vec{x}$ on the sensor image plane is shown to have the general form

$$g(u, v) = h_o(u, v, \vec{x}) \tag{2-2}$$

where u, v are the sensor coordinates

In section 4, a measurement model is shown to obey an equation of the form

$$I(u, v) = \int_{t-\Delta}^{t} dy_i \tag{2-3}$$

where I(u,v) is the intensity function at (u,v),

y_i is the i^{th} component of vector $\vec{y}$ and $\vec{y}$ is defined by

$$d\vec{y} = \vec{h}(\vec{x})dt + d\vec{z} \tag{2-4}$$

$$\vec{h}(\vec{x}) = [h_o(u_1, v_1, \vec{x}), h_o(u_2, v_2, \vec{x}), h_o(u_3, v_3, \vec{x}) \ldots]^T \tag{2-5}$$

with u_i, v_i being the u,v coordinates of the i^{th} pixel.

Equations (2-1) and (2-4) formally define the system in terms of stochastic differential equations, where the tracker synthesis problem is to find a near optimal estimate of $\hat{\vec{x}}$ based on measurements $\vec{y}$ and the performance estimation problem is to estimate or bound the performance of such estimates.

2.2 STOCHASTIC DIFFERENTIAL EQUATIONS

Analysis of the tracking problem in terms of stochastic differential equations requires the introduction of special techniques, since the differential of the random processes involved does not exist in the sense of ordinary differentials. The formulation that provides meaning to these special differentials is generally referred to as Itô calculus and the interested reader is referred to any of the general texts on this subject listed in the bibliography, e.g., McGarty (2). Here, the basic concepts useful in tracker analysis are summarized to provide guidance for further study.

The first concept is that of independent increments. Basically, this concept means that increments of the process over non-overlapping (time) intervals are independent. Formally, the definition is as follows:

$$\text{Let for } t_j < t_i \leq t_m < t_n$$

$$v = x(t_i) - x(t_j) \qquad (2\text{-}6)$$

$$u = x(t_n) - x(t_m)$$

Then x(t) is an independent increment process if v and u are independent random variables.

The importance of this concept lies in the fact that two well known random processes that are directly applicable to modeling the tracker problem satisfy these conditions. These are the Wiener and Poisson processes.

The Wiener process is useful both for modeling noise in measurements model and disturbances in the dynamical model. This is the most common process

used in estimation theory, and has the following formal definition:

$x(t)$ is a Wiener process if it has independent increments, and

$$x(0) = 0$$

$$p[(x(t) - x(s)) \leq \lambda] = \frac{1}{\sigma\sqrt{2\pi(t-s)}} \int_{-\infty}^{\lambda} e^{-\eta^2/(2\sigma^2(t-s))}\, d\eta \qquad (2\text{-}7)$$

for

$$t \geq s$$

As a consequence, then, this process has the following important properties:

$$E[x(t)] = 0$$

$$E[x^2(t)] = \sigma^2 t$$

$$E[x(t)\, x(s)] = \min(t, s)$$

and, if

$$\Delta x = x(t + \Delta t) - x(t) \qquad (2\text{-}8)$$

then

$$E[\Delta x^2] = \sigma^2 \Delta t \qquad (2\text{-}9)$$

The Poisson process (2) is useful in modeling step changes in the system state caused by execution of maneuvers at random times. The formal definition of a Poisson process is as follows:

$x(t)$ is a Poisson process if it has independent increments, the sample function $x(t,w)$ is a step function increasing with jump 1,* and:

---

* A Poisson process can be generalized to have several jump values a_i each with a characteristic probability of occurrence P_{a_i}

$$x(0) = 0$$

$$p[x(t) - x(s) \geq k] = e^{-\lambda|t-s|} \frac{(|t-s|)^k \lambda^k}{k!}$$

where $\lambda > 0$ (2-10)

Also

$$x(t-) \neq x(t) = x(t+) \tag{2-11}$$

i.e. continuous on the right.

The special properties of these processes lead to a requirement to extend the definitions of differential and integral stochastic processes. As shown above, e.g., the second moment of the Wiener process is of the order dt, not dt^2 so that higher order terms must be retained in any expansions. This property also results in the process not being of bounded variation, so differentials do not exist in the ordinary sense. The solution to this problem was developed by Itô and is illustrated here by the Itô rule for differentials, presented here in the scalar form.

Let

$$y = f(x) \tag{2-12}$$

where

$$dx = g(x)\, dt + dw \tag{2-13}$$

and

$$E[dw\, dw] = W dt \tag{2-14}$$

The problem is to find the differential dy. The approach is to perform an expansion through second order terms.

Then

$$\begin{aligned} \Delta Y &= f(x + \Delta x) - f(x) \\ &\cong \frac{df}{dx}\Delta x + \frac{1}{2}\left(\frac{d^2f}{dx^2}\right)\Delta x^2 \\ &\cong \frac{df}{dx}\Delta x + \frac{1}{2}\frac{d^2f}{dx^2}\left(g(x)\Delta t + \Delta w\right)^2 \end{aligned} \tag{2-15}$$

Retaining only first order terms in Δt results in

$$\Delta Y \cong \frac{df}{dx}\Delta x + \frac{1}{2}\frac{d^2f}{dx^2} W \Delta t \tag{2-16}$$

which, by taking limits, results in

$$dy = \left(\frac{df}{dx}\right)dx + \frac{1}{2}\left(\frac{d^2f}{dx^2}\right) W\, dt \tag{2-17}$$

Note that dw^2 was replaced by its expectation, since with probability 1,

for any finite t. $$\int_0^t (dw)^2 \to Wt$$

With this understanding of stochastic differentials, the stochastic differential equation formulation of the tracker problem given formally in section 2.1 has meaning. The problem, then, is from observing the measurement $\vec{y}$ estimate the probability density of the state vector $\vec{x}$. Then the minimum mean square error (MMSE), or maximum aposteriori (MAP) estimate of $\vec{x}$ can be determined. In addition, from the probability density of $\vec{x}$, the error can be estimated.

In developing a model of a system in terms of stochastic differential equations, an additional effect needs to be considered. Suppose the system can be represented in the form

$$dx = f(x,t)\,dt + g(x,t)\,dn_B \tag{2-18}$$

where $n_B(t)$ is integrated white noise of bandwidth B and w(t) is a Wiener process satisfying

$$\lim_{B\to\infty} \int (dn_B - dw) = 0 \tag{2-19}$$

Then

$$\begin{aligned}
\int (dx - f(x,t)\,dt) &= \lim_{B\to\infty} \int \left(g(x,t) + \frac{1}{2}\,\frac{\partial g}{\partial x}\,dx\right) dn_B \\
&= \lim_{B\to\infty} \int \left(g(x,t) + \frac{1}{2}\,\frac{\partial g}{\partial x}\,g(x,t)\,dn_B\right) dn_B \\
&= \lim_{B\to\infty} \int \left(g(x,t) + \frac{1}{2}\,\frac{\partial g}{\partial x}\,g(x,t)\,dw\right) dw \\
&\quad - \int g(dw - dn_B) - \frac{1}{2}\int \frac{\partial g}{\partial x}\,g\,(dw^2 - dn_B{}^2) \\
&= \int \left(g(x,t) + \frac{1}{2}\,\frac{\partial g}{\partial x}\,g\,dw\right) dw - \frac{1}{2}\int \frac{\partial g}{\partial x}\,g\,dt
\end{aligned} \tag{2-20}$$

The last term is referred to as the Wong-Zakai correction, so in the limit as $B \to \infty$, x obeys the differential equation

$$dx = \left[f(x,t) - \frac{1}{2}\,g\,\frac{\partial g}{\partial x}\right] dt + g(x,t)\,dw \tag{2-21}$$

2.3 SOLUTION OF THE ESTIMATION PROBLEM

As stated in section 2.2, the basic estimation problem is to find the probability density function P of the state $\vec{x}$ given the observation or measurement $\vec{y}$. There are two useful approaches. Kushner has developed a complete solution for P in terms of a stochastic partial differential equation (ref (43)) and the Bucy representation theorem can be used for discrete simulation of results on a digital computer. The Kushner solution is given here for an estimation problem with generalized Poisson process in addition to Wiener processes. The formal definition of the basis estimation problem is given in Table 2-1, along with Kushner's solution for the probability density function P.

This equation is a nonlinear stochastic partial differential equation and has no general solution. Thus the problem is in finding tractable analytic means for dealing with it. A simple graphical illustration of the terms in the Kushner equation, as shown in Table 2-2, helps to provide an insight into the nature of this problem. The first term, A (y), is a matched filter term

TABLE 2-1. FORMULATION AND FORMAL SOLUTION OF THE OPTIMAL ESTIMATION PROBLEM USING STOCHASTIC DIFFERENTIAL EQUATIONS

Let $dx = f(x,t)dt + dw + dn$
$dy = h(x,t)dt + dz$

w, z Wiener process with $E[dw\ dw] = Wdt$, $E[dz\ dz] = Zdt$
dn Poisson process with rate $\vec{\lambda}$ and transition density P_{a_i}

Let $P \equiv P(x,t)$ represent probability density of x(t) at time t

Then

$$dP = A(y) + B(P)dt + C(P)dt$$

Where

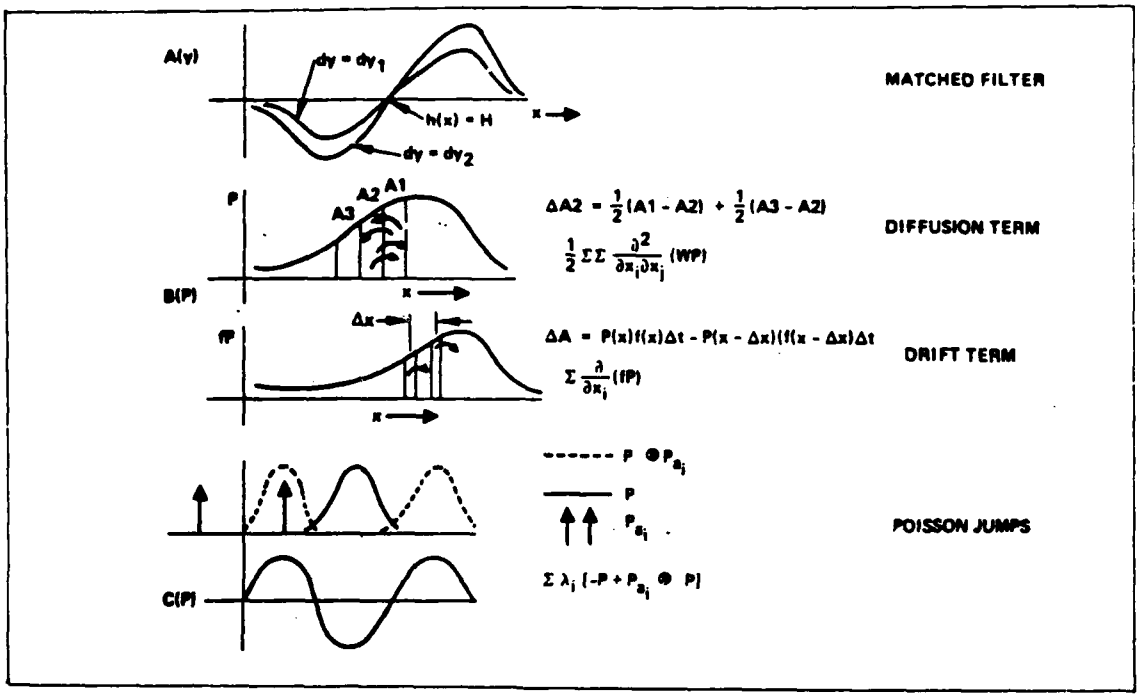
$$A(y) = P(dy - Hdt)Z^{-1}\ (h - H)\ \text{"Matched Filter"}$$

$$B(P) = \sum \frac{\partial}{\partial x_i}\ (f_i P) + \frac{1}{2} \sum \sum \frac{\partial^2}{\partial x_i \partial x_j}\ (WP)\ \text{"Drift-Diffusion"}$$

$$C(P) = \sum \lambda_i\ [-P + P_{a_i} \circledast P]\ \text{"Abrupt Change"}$$

$$H = \int h(\vec{x})\ P(\vec{x},t)d\vec{x} = E[h]$$

TABLE 2-2. GRAPHICAL ILLUSTRATION OF TERMS IN THE KUSHNER EQUATION



modified by the probability function $P(\bar{x},t)$. This is illustrated in Table 2-2 for two different measurements dy and dy_2. The roll-off, as the observation deviates from H, the expected value of h (x), is due to the P $(\bar{x},t)$ modification.

The second term B(p) is a combination drift and diffusion. This is a noise effect and is represented by a second order partial derivative, since it is a relative effect. For example, as shown in the table, the net change in the A_2 region is the resultant of the diffusion from A, into A_2 minus the diffusion from A_2 into A_1 and similarly for regions A_2 and A_3. The drift represents the fact that f $(\bar{x},t)$ will cause the probability distribution to have a net shift as shown in the figure.

The third term is the Poisson jump terms. Referring, again, to the table, the arrows represent the probability of a jump of magnitude equal to the x coordinate of the arrow. This is convolved with the old probability density (solid line), resulting in a new probability density (one of the dashed lines).

This is then modified by the probability λ_i that a jump will occur in that time interval, resulting in the Poisson jump contribution to the Kushner equation.

In addition to the continuous time solution represented by the Kushner partial differential equation, there is a discrete time, integral equation formulation (see Table 2-3), and its solution is equivalent to the solution of Kushner's equation. This discrete solution is more useful for digital simulation, which by it's very nature is discrete. This solution is based on the representation theorem.

Representation Theorem

The representation theorem first proven by Bucy, expresses the conditional densityin an integral which could be used to derive results which would be difficult or impossible using Kushner's equations. In fact, Kushner's equations can be derived from this theorem. Using the representation theorem, we get for $\vec{x}$ and $\vec{y}$ defined as in section 2, and for

$$\lambda = 0$$

$$P_x(u,t|y(\tau),\ \tau \le t) = \frac{E[e^{Ht}|x(t)=u]}{E[e^{Ht}]}\ P_{\vec{x}}(\vec{u},t) \qquad (2\text{-}22)$$

with E[ ] denoting expectation and

$$H_t = \int_{-\infty}^{t} h^T(x(\eta),\eta)\ Z^{-1}(dy(\eta) - \frac{1}{2}\ h(x(\eta),\ \eta)\ d\eta) \qquad (2\text{-}23)$$

The compactness and elegance of equations (2-22) and (2-23) allows this expression for the state probability density to be used effectively in proofs related to the stochastic filtering problem (see (2)).

TABLE 2-3. DISCRETE TIME INTEGRAL FORMULATION

$$\hat{x}(K) = \hat{f}(\hat{x}(K-1)) + \hat{W}_1(K-1)$$

$$\hat{z}(K) = \hat{h}(\hat{x}(K-1)) + \hat{W}_2(K-1) \qquad W_1, W_2 \text{ GAUSSIAN}$$

$$P(\hat{U}|\hat{z}(i), i \leqslant K) \equiv J_K(\hat{U})$$

THEN

$$J_K(\hat{U}) = \frac{L_K(\hat{U})}{\int L_K(\hat{U})d\hat{U}}$$

$$L_K(\hat{U}) \equiv \text{LIKELIHOOD OF } \hat{U}$$

$$L_K(\hat{U}) = N(\hat{z}(K) - \hat{h}(\hat{U}), \underset{\sim}{R}) \times \int N((\hat{U} - f(\hat{V})), \underset{\sim}{Q})\, J_{K-1}(\hat{V})\, d\hat{V}$$

$$\underset{\sim}{R} = E[\hat{W}_2(K)\, \hat{W}_2(K)]$$

$$\underset{\sim}{Q} = E[\hat{W}_1(K)\, \hat{W}_1(K)]$$

$$N(\hat{A}, \underset{\sim}{B}) = EXP\,[(-1/2) \cdot (\hat{A}^T \underset{\sim}{B}^{-1} \hat{A})]$$

Although neither the integral nor differential form of the solution can be solved in general, there are some special cases where a solution can be obtained. For the case of linear dynamics and linear measurements with Gaussian noise, it reduces to the familiar Kalman-Bucy filter. Similarly, many non-linear systems may be approximated by a linearization of the form

$$\begin{aligned} \vec{f}(\vec{x},t) &= \vec{f}(\vec{x}^*,t) + A(t)(\vec{x}-\vec{x}^*) \\ \vec{h}(\vec{x},t) &= \vec{h}(\vec{x}^*,t) + C(t)(\vec{x}-\vec{x}^*) \end{aligned} \tag{2-24}$$

and the equation will reduce to the extended Kalman-Bucy filter for estimating $\vec{x}-\vec{x}^*$. Explicit formulations for these cases can be found in (3).

2.4 PERFORMANCE BOUNDS AND APPROXIMATIONS

For non-linear problems, a closed form solution of the Kushner equation cannot be produced, in general. The problems of calculation of the optimal non-linear estimator, or even the calculation of the performance of simple sub-optimal estimators, are computationally intractable, since both would require infinite dimensional calculation [22]. The principal problem in calculating performance is that the propagation of the second moment includes not only the past time history of the second moment, but also of all higher moments. This results in an infinite chain of higher order moments that must be propagated in order to establish mean square error performance. This behavior is in dramatic contrast to the situation with linear systems where the second moment, or mean square, error propagates by a Riccati type differential equation, and higher order moments have no effect on performance evaluation. In order to circumvent the problems inherent in the non-linear tracker estimation problem, two related analytic techniques that are both practical and mathematically rigorous can be used: performance bounds and approximation by linearization.

Performance bounds are generated by making controlled approximations that guarantee that the resulting performance estimate is either less than (lower bound) or greater than (upper bound) the performance of the optimal estimator, or for any candidate suboptimal estimates. Such bounds can always be found, but the analytical trick is to make the approximations judiciously so that the resulting bounds are in some sense "tight" and non-trivial. A tight bound can be assured if the difference between the upper and lower bounds is small.

Good performance bounds can be extremely valuable to the system designer. If the lower bound of a candidate suboptimal design is close to the lower bound on optimal performance, he knows he has done his job well and further refinements or improvements may not be cost effective or may be futile. Alternately, if the required performance is lower than the lower bound on the optimal system, the futility of the system design problem becomes apparent. Upper bounds are useful in that if the upper bound of a candidate design is lower then the system requirements, then it is not necessary to seek further improvements in design. The performance bounds also provide the all-important functional relationship between system parameters and performance. This allows the designer to perform tradeoffs on the parameter requirements to insure a cost effective system. For the specific application to tracker systems, these bounds can be obtained for both the mean square tracking accuracy and the probability of losing lock, both important figures of merit for trackers.

The second useful analytic technique is to approximate the non-linear estimation problem with a linear problem, through a linearization technique. This approach also provides a functional relationship between system parameters and performance, and provides an estimate of the actual performance. This technique can be used in conjunction with the performance bound technique. If the bounds are tight and the performance of the extended Kalman-Bucy filter for the linearized system falls between the upper and lower bounds, then the performance of the linearized system must be close to the optimal non-linear estimator. This is often the case and then the linearization approach provides the design solution. Linearization is also important in that it is often used as a step in computing performance bounds, as shown below. An example of the use of linearization is given in section 5.

Two general techniques have been identified and investigated for computing mean square error performance bounds for the non-linear estimation problems. Each of these approaches relates performance bounds from the optimal non-linear system to performance of related linear systems.

The first approach, the Bobrovsky-Zakai bound (23), is based on an infinite-dimensional extension of the Van Trees version of the Cramer-Rao bound (42). This extended bound is used to define a linear system related to the original non-linear system, but not a direct linearization, that has the property that the optimal solution of the linear system has an MMSE that forms a lower bound for the optimal non-linear estimation error. An example of this approach for a simple one-dimensional problem is presented in section 5, with details of the derivation in appendix E.

The second approach is termed the cone-bounded approach, and it provides both upper and lower bounds on the performance of the optimal non-linear estimator in terms of the performance of the associated linearized system and certain non-linearity parameters that are representative of the degree of the non-linearities. These bounds apply to a restricted class of non-linear systems that satisfy a uniform Lipschitz condition of the form

$$\begin{aligned} ||f(x+\Delta x,t) - f(x,t) - A(t)\ \Delta x|| &\le a(t)||\ \Delta x|| \\ ||h(x+\Delta x,t) - h(x,t) - C(t)\ \Delta x|| &\le c(t)||\ \Delta x|| \end{aligned} \qquad (2\text{-}28)$$

Although this approach is restricted to systems that satisfy (2-28), the approach is still useful since similar Lipschitz conditions are usually needed to assure unique solutions to ordinary non-linear differential equations. Since physical systems are generally of this type, this approach will be useful in practical applications.

The result for the lower bound has been extended in this project to give the following result. Let the linearized system be

$$dx = Axdt + dw$$
$$dy = Cxdt + dZ \qquad (2\text{-}29)$$

and let $\hat{P}$ be the error covariance matrix associated with the optimal filter for this system. Then, the lower bound for the non-linear system covariance, Q, is given by

$$Q \geq (1 - r(t))\ \hat{P}(t) \qquad (2\text{-}30)$$

where r(t) is the unique non-negative root of the equation

$$(1 - r(t))\ e^{r(t)} = e^{-d(t)} \qquad (2\text{-}31)$$

and

$$d(t) = \int_0^t \left[a^2(S)\ W^{-1}(S) + c^2(S)\ Z^{-1}(S)\right]\ T_r\left[\hat{P}(S)\right] dS \qquad (2\text{-}32)$$

where $T_r(\hat{P})$ = Trace of $\hat{P}$.

This is a recent improvement [44] over the original result [22] where d(t) was computed using the trace of the upper bound on performance. Thus, the lower bound is both tighter and much simpler to evaluate.

Note that if α and β are defined as

$$-\ln[\alpha(t)] = \int_0^t a^2(S)\ W^{-1}(S)\ Tr\ \hat{P}(S)\ \ dS$$
$$-\ln[\beta(t)] = \int_0^t c^2(S)\ Z^{-1}(S)\ Tr\ \hat{P}(S)\ dS \qquad (2\text{-}33)$$

then r(t) satisfies

$$(1-r(t))\ e^{r(t)} = \alpha(t)\ \beta(t) \qquad (2\text{-}34)$$

Thus, α and β are measurements of the degree of non-linearity of the dynamics and measurement equations, respectively.

The principal problem in using this approach is in maintaining a tight bound in the Lipschitz condition (equation 2-28). If the $\alpha(t)$ or $c(t)$ functions are too large, then both the upper and lower bounds will quickly degenerate into trivial bounds. This is a consequence of equation (2-32), which shows that if these functions are too large, then $d(t)$ will become cone bounded, forcing $r(t)$ to approach 1. The lower bound then, e.g., becomes 0 from equation (2-30).

2.5 BOUNDS ON PROBABILITY OF LOSING TRACK

The techniques discussed in Section 2.4 applied to finding bounds on the accuracy of tracking, and as such, are not applicable to the problem where the tracker has completely lost track. Thus, an additional performance parameter is needed. This parameter is the probability of losing track in some time interval. This is analogous to the situation in the ATAC Autocuer Modeling Analysis [30] where, in addition to accuracy in locating an object, the probability of totally missing the object, called the probability of anomaly, was needed to provide complete performance analysis.

The probability of losing track is relative to the concept of non-linear stability. This concept is illustrated in Figure 2-1. Basically, the concept involves answering the question: Given a non-linear system with random inputs and outputs, what is the bound on the probability that the random process output is greater than some threshold in some interval, given its initial value? Here, an approach to finding an upper bound to the probability of losing track is presented.

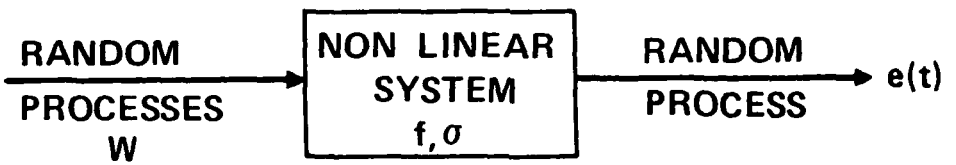
A more in-depth discussion of this method can be found in Kushner [1]. The basic method can be applied to any stochastic process which satisfies an equation of the form

$$d\vec{X} = \vec{f}(\vec{X},t)dt + \sigma(\vec{X},t)d\vec{w} \qquad (2\text{-}35)$$

where $\vec{X}$ is a n-dimensional state vector,

$\sigma(\vec{Z},t)$ is a nxm matrix

$\vec{w}$, a m-dimensional normalized Wiener process.



QUESTION – WHAT IS BOUND ON PROBABILITY

$\|e(t)\| \geq E$ IN INTERVAL $0 \leq t \leq T$

GIVEN $\|e(o)\| = e_o$

ANSWER

$$p\,(\text{SUP}_{0 \leq t \leq T} \|e(s)\| \geq E) \leq 1 - \left(1 - \frac{e_o}{E}\right) e^{-\Phi T/E}$$

WHERE $\Phi_T = \int_o^T \varphi(s)\, ds$

AND $\varphi(\cdot)$ IS ANY FUNCTION φ SATISFYING

$A\|e\| \leq -\mu\|e\| + \varphi(t)$ WHERE A IS THE DIFFERENTIAL GENERATOR

FIGURE 2-1. NON-LINEAR STABILITY

For applications to the tracker problem, the state variable in (2-35) will generally be taken to be the error process

$$\vec{e} = \vec{x} - \hat{\vec{x}}$$

where $\vec{x}$ is the target location and $\hat{\vec{x}}$ is the target location estimate. In general, it will not be possible to write the error process in the state equation form

$$d\vec{e} = \vec{f}(\vec{e},t)dt + \sigma(\vec{e},t)dw \ . \qquad (2\text{-}36)$$

However, a linearization of

$$d\vec{e} = d\vec{x} - d\hat{\vec{x}}$$

about the state estimate $\hat{\vec{x}}$ will result in an equation of form (2-36). This is necessary in order to apply the methods in [1].

For the tracking problem, one will be interested in the probability that

$$||\vec{e}|| \geq m \ .$$

This probability is bounded via methods in [1] by first applying the differential generator A to $||\vec{e}||$

where $$A = \sum_i f_i(\vec{e},t) \frac{\partial}{\partial e_i} + \sum_i \sum_j (\sigma \sigma^T)_{ij} \frac{\partial^2}{\partial e_i \partial e_j}$$

f_i is the i^{th} element of the vector $\vec{f}$ and $(\sigma \sigma^T)_{ij}$ is the i,j element of the matrix $\sigma \sigma^T$. The differential generator A can be treated as the expected value of the time derivations of its argument. If positive μ and ϕ_t can be found such that

$$A||e|| \leq -\mu ||e|| + \phi(t),$$

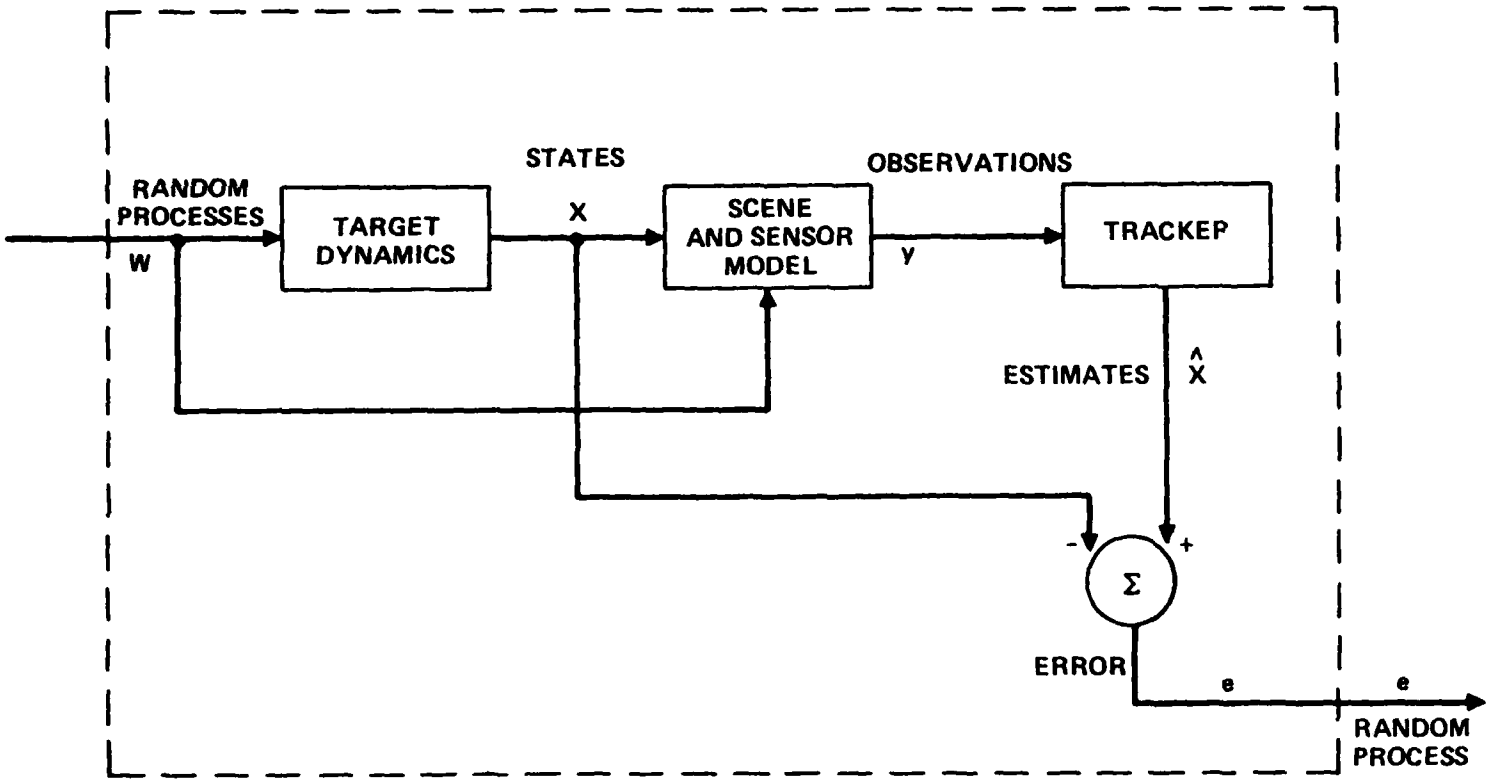
then the upper bound is given by

$$P\left(\sup_{0 \leq s \leq t} ||\vec{e}(s)|| \geq m \,\middle|\, ||\vec{e}(0)|| = k \right) \leq 1 - \left(1 - \frac{k}{m}\right) \exp\left(-\frac{1}{m} \int_0^t \phi(s)\, ds \right)$$

In effect the μ can be looked at as a degenerative feedback term on the error norm $||e||$ and $\phi(t)$ can be modeled as the degree to which the tracker departs from an idealized one pole feedback system.

The overall approach to using this concept of non-linear stability to compute the probability of losing lock is illustrated in Figure 2-2. The basic problem is to find a bound on the random process e, which is the difference between the estimate of the state and the true state. Having a bound on this error provides a means for bounding the probability of losing lock, and the method discussed above provides one means of doing this.

The key is an upper bound on the probability that in the time interval (0,t) the error will exceed a specified value, given that the error is known at time 0. The procedures applies generally to any suitable function G(e), not just the norm. Selecting this function (similar to selecting a Liaponov function for other stability applications) is a key step in the procedure as is selecting ϕ_t which can be quite arbitrary. For linear systems, some insight into the nature of the problem must be used since these parameters basically relate to the degree of non-linearity. The procedure can always be applied, some μ and $\phi(t)$ can always be found, however, if they are not selected carefully, the bound may prove to be trivial. Note that since the process is applied to the error in the estimation, all normalizing factors previously identified in the performance analysis of trackers also apply here.



- MODEL TARGET-SCENE-TRACKER AS SINGLE STOCHASTIC PROCESS
- BOUNDING PROBABILITY OF $\|e\|$ BOUNDS PROBABILITY OF LOSING LOCK

FIGURE 2-2. USE OF NON-LINEAR STABILITY

2.6 USE OF THE ANALYTIC TECHNIQUES

The overall approach to using the techniques discussed in the previous sections is summarized in Figure 2-3. The first step is to define the particular tracker problem with the result being a set of system models, including a dynamics model and a measurement model. Then the performance bounds and approximation techniques are used to delimit the parameter space. Then for that parameter space, Kushner's equation can be solved numerically to obtain an exact solution of the bounds for comparison, adjusting the delimited parameter space as necessary to achieve the desired results. Then algorithms can be defined that approximate the solution to Kushner's equation, representing perhaps the actual tracker implementation. The algorithms can then be compared to the exact Kushner solution to indicate how close to optimal they are. The procedure is iterated until acceptable results are found.

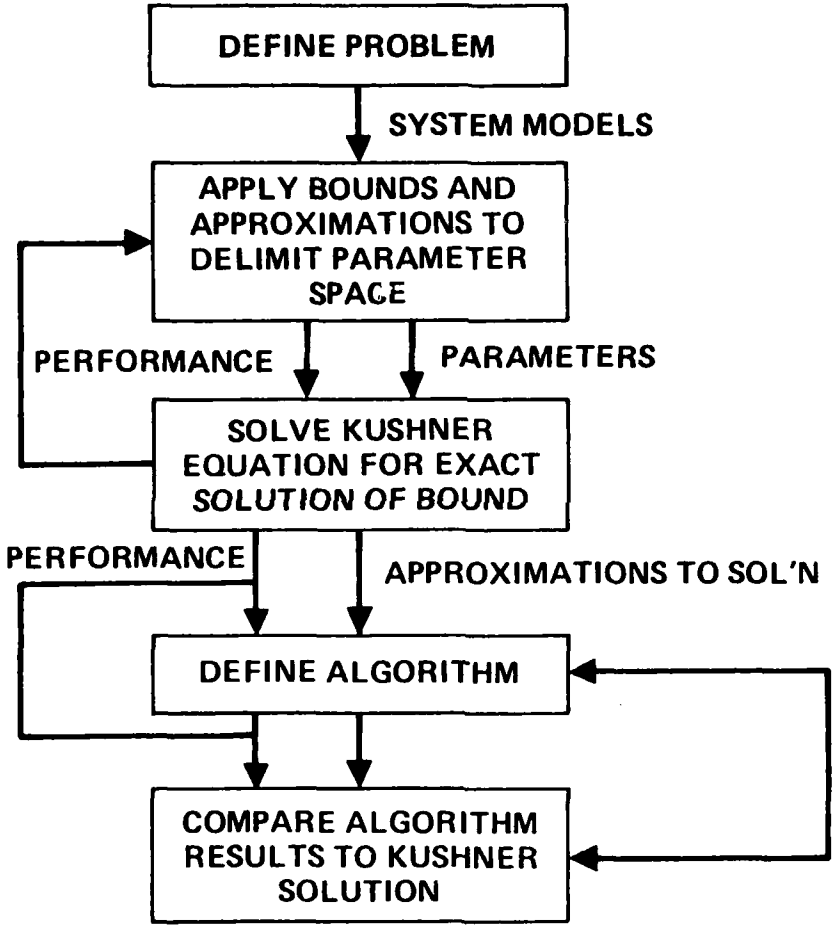


FIGURE 2-3. USE OF THE ANALYTIC TECHNIQUES

SECTION 3. TARGET DYNAMICS MODEL

In this section, a model is developed to describe target and target dynamics for a three-dimensional target moving with 6 degrees of freedom (3 translational, 3 rotational). This model is specialized to the primary case of interest in this study: A tactical vehicle (tank) where motion is constrained by local terrain and maneuvers are constrained by available controls through interaction with the local terrain. The target contributes to the state vector and dynamics stochastic equation. A simplified target model, approximated by a rectangular solid, is defined and the basic features of the resulting two-dimensional image projected into the image plane are discussed in Appendix B. This model, both target and dynamics, when combined with the scene model developed in section 4, provides the stochastic differential equation model for the tracker estimation problem. Also presented in Appendix D are typical numerical values for the parameters that determine the basic maneuvering capability of tanks.

3.1 THREE-DIMENSIONAL TARGET MODEL

The geometry for the three-dimensional target model as related to the sensor image plane is shown in Figure 3-1 for a rectangular solid target. The concept involved is, however, applicable to arbitrary three-dimensional target shapes. The basic approach is to define the orientation of the target in terms of the transformation from target body fixed coordinates to sensor coordinates. The three angles necessary to define this transformation, along with the target position relative to the sensor, provide a mapping of the three-dimensional target shape to a two-dimensional apparent shape via sensor projection model developed in section 4.

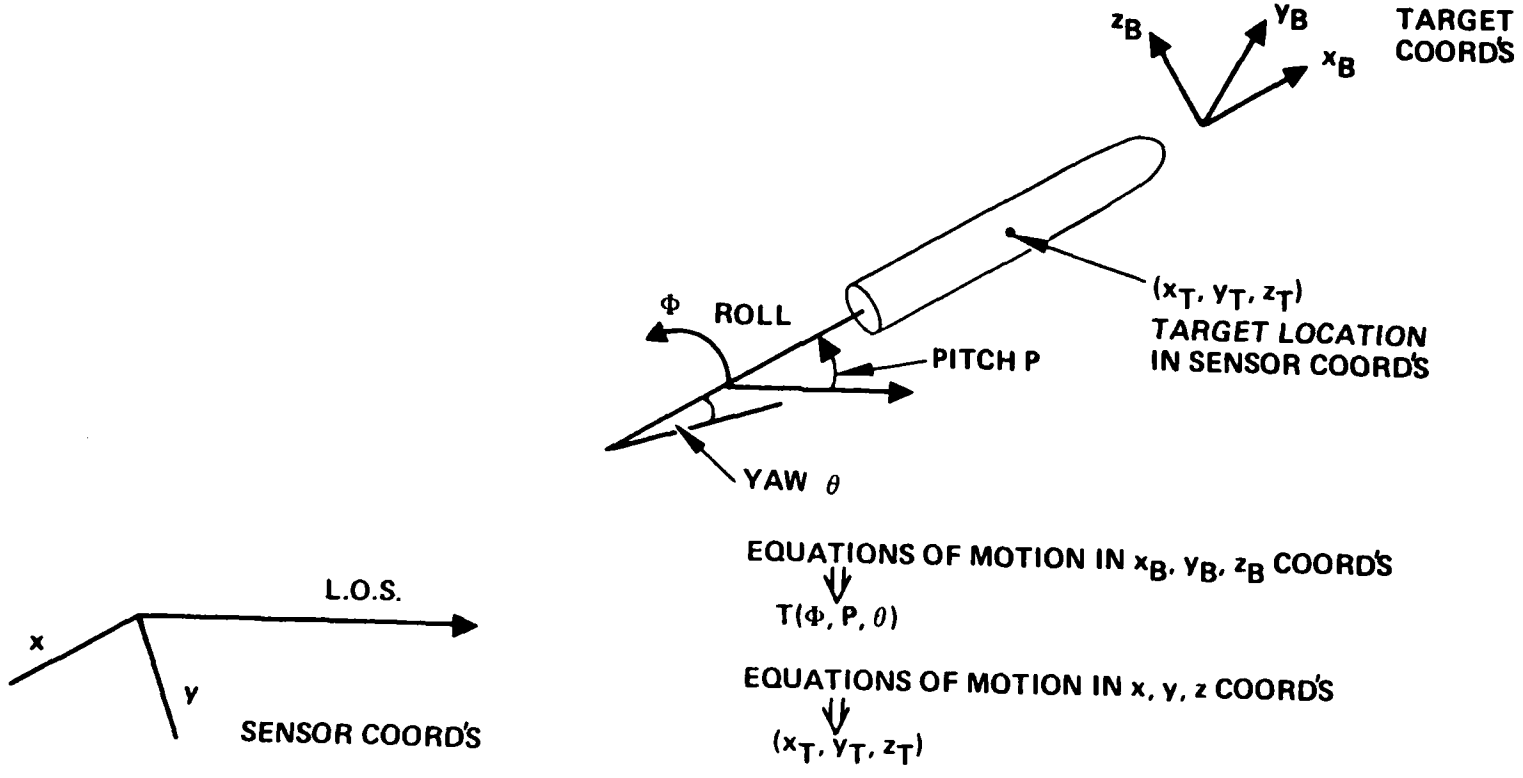


FIGURE 3-1. TARGET GEOMETRY

The target body coordinate axes x_b, y_b, z_b are defined so that x_b lies along the direction of the principal force used for speed control, z_b along the principal axis for turning or direction changes, and y_b forming a right-handed coordinate system. The two-dimensional shape is determined by the aspect angle θ (yaw) and the pitch angle P, while the roll angle ϕ determines the orientation of the shape in the image plane. The transformation from body coordinates to sensor local vertical coordinates (x,y,z) is determined by the following three rotation transformations.

1). Target Yaw

$$[A]_3 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2). Target Pitch

$$[P]_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos P & \sin P \\ 0 & -\sin P & \cos P \end{bmatrix}$$

3). Image Roll

$$[\phi]_2 = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix}$$

The complete transformation is then given by

$$T = [\phi]_2 [P]_1 [\theta]_3$$

The remaining three parameters for the target model are the x_T, y_T, z_T position of the target relative to the sensor.

3.2 DYNAMICS MODEL FOR A TANK

The basic model for a maneuvering tank is depicted in Figure 3-2. The salient features of this model are quite simple. First, the tank is confined to follow the local terrain, excluding momentary bump effects. Second, the only possible maneuvers are linear acceleration along the hull major axis and a turn around what is basically the turret axis, and these are only possible through interaction with the local terrain. Thus, the possible direction of a maneuver depends on the tank orientation and the local terrain slope. In addition, tank type restricts the possible maneuvers, e.g., maximum turn rate, speed, acceleration. Knowledge of maneuver capability is essential for obtaining a good dynamic model for filtering and prediction. Similar techniques are used for other vehicle types.

The basic equations of motion for this tank can be written in the form

$$\ddot{\vec{r}} = \frac{F}{m}\vec{J} - \frac{K_1}{m}\dot{\vec{r}}$$

$$\ddot{\theta} = \frac{\tau}{I}1 - \frac{K_2}{I}\dot{\theta}$$

where

F is motor or braking force

K_1 and K_2 are damping constants

τ_1 is the torque force causing vehicle turn

$\dot{\theta}$ is the turning rate

and

m and I are the tank mass and moment of inertia about the turning axis.

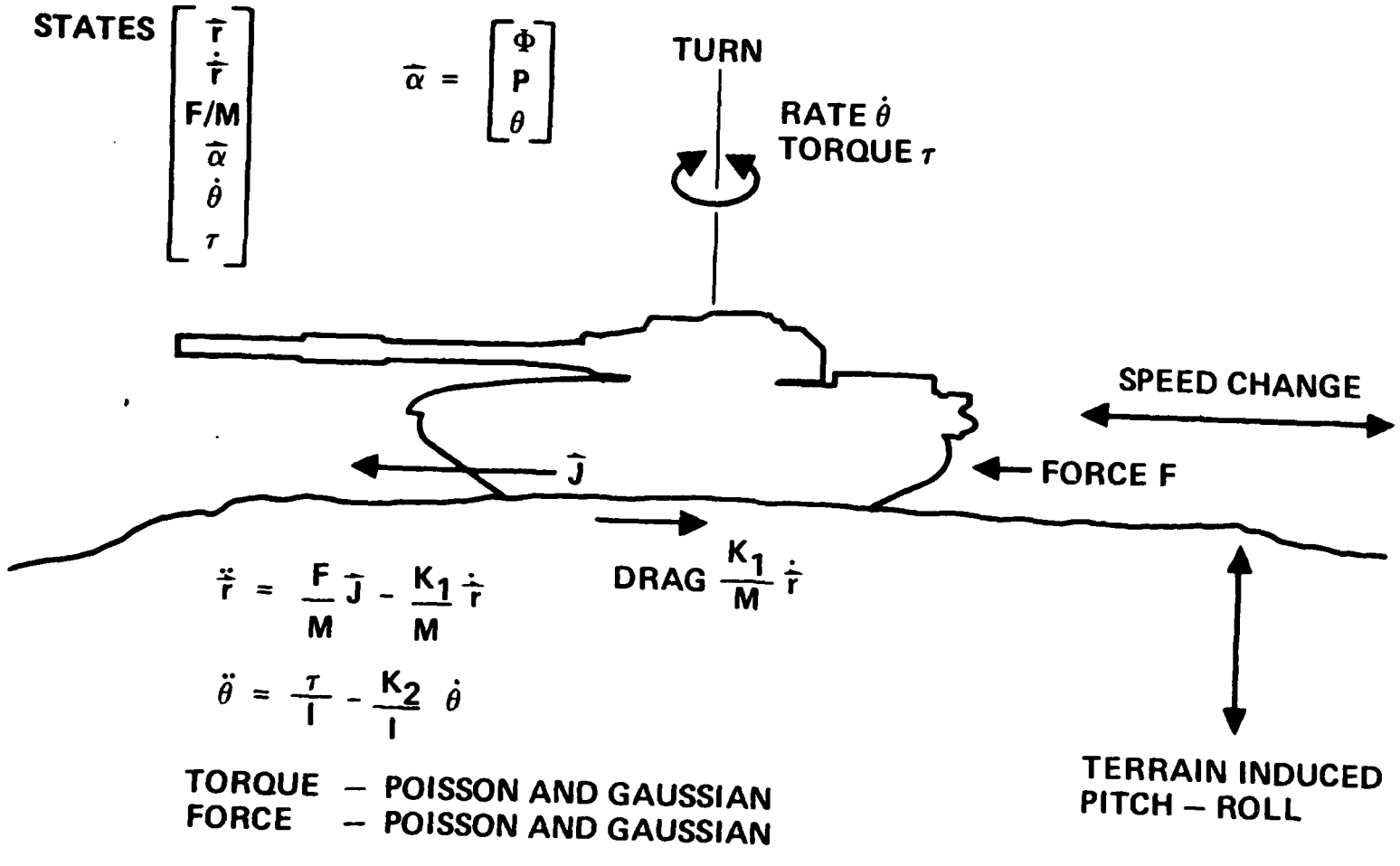


FIGURE 3-2. SAMPLE TARGET MODEL

From the definitions of the transformation from target to sensor coordinates above, the components of the vector $\vec{J}$ can be defined as

$$\vec{J} \quad \begin{bmatrix} \text{Cos } \phi \text{ Cos } \theta - \text{Sin } \phi \text{ Sin } \theta \text{ Sin P} \\ - \text{Sin } \theta \text{ Cos } \phi \\ \text{Cos } \theta \text{ Sin } \phi + \text{Cos } \phi \text{ Sin } \theta \text{ Sin P} \end{bmatrix}$$

when expressed in sensor local vertical coordinates.

There are two types of disturbances associated with this dynamic model. The first type termed controlled disturbances are the result of the maneuvering by altering the force F and torque. The second type is the uncontrolled effects induced by terrain variations, which are primarily small disturbances on the pitch and roll of the target. The model for the torque disturbance is shown in Figure 3-3, and consists of two types: Generalized Poisson which models random turning of the tank under operator control and white Gaussian representing terrain induced effects.

The target portion of the dynamics state equations then can be written as follows:

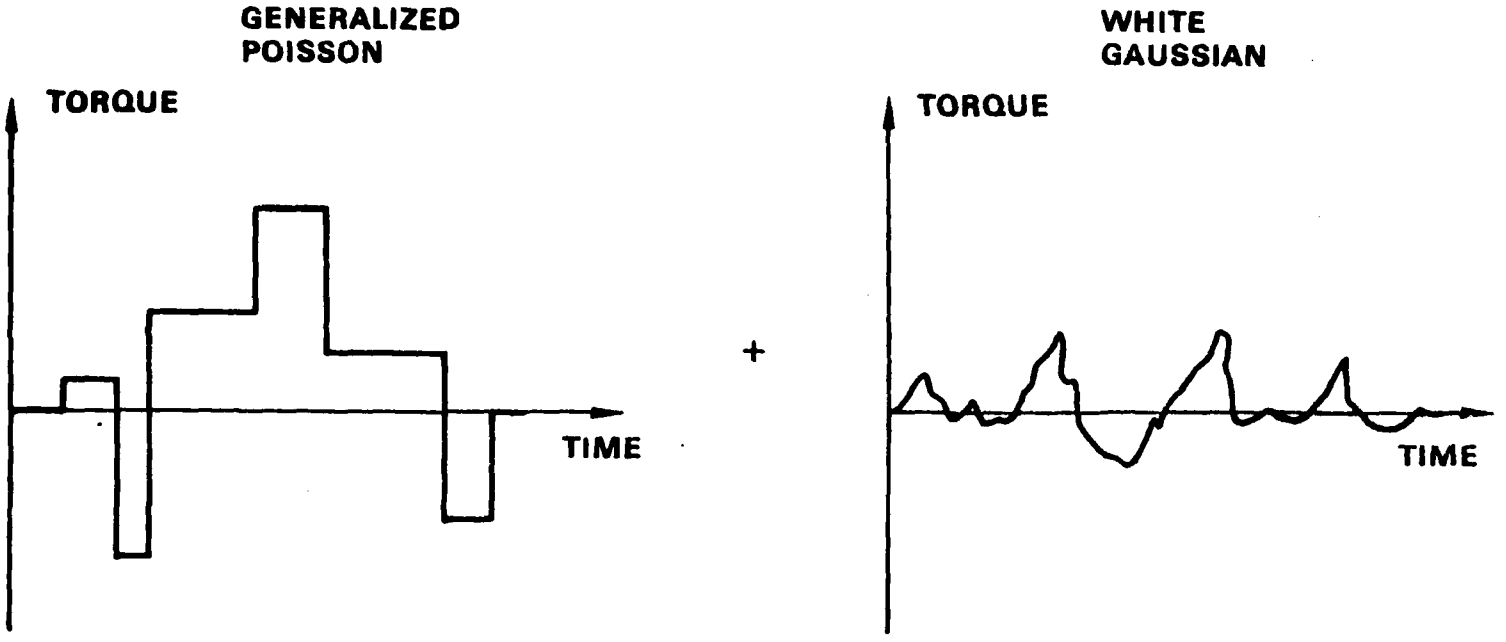


FIGURE 3-3. TORQUE MODEL

$x_1 = x$
$x_2 = y$
$x_3 = z$ } POSITION

$x_4 = \dot{x}$
$x_5 = \dot{y}$
$x_6 = \dot{z}$ } VELOCITY

$x_7 = F/m$ DRIVING FORCE

$x_8 = \phi$
$x_9 = P$
$x_{10} = \theta$ } ANGLES

$x_{11} = \dot{\theta}$ ANGLE RATE

$x_{12} = \tau$ Torque

Using these definition along with the equations of motion and the disturbances gives the following set of state equations

$$dx_1 = x_4 dt$$

$$dx_2 = x_5 dt$$

$$dx_3 = x_6 dt$$

$$dx_4 = \left[X_7(\cos X_8 \cos X_{10} - \text{Sin } X_8 \text{ Sin } X_{10} \text{ Sin } X_9) - \frac{K_1}{m} X_4\right] dt$$

$$dx_5 = \left[X_7(-\text{Sin } X_{10} \cos X_8) - \frac{K_1}{m} X_5\right] dt$$

$$dx_6 = \left[X_7(\cos X_{10} \text{ Sin } X_8 + \cos X_8 \text{ Sin } X_{10} \text{ Sin } X_9) - \frac{K_1}{m} X_6\right] dt$$

$dx_7 = dF_1 + dF_2$ - F_1 Wiener, F_2 Poisson processes

$dx_8 = d\phi_1$ - ϕ_1 Wiener driving process on roll

$dx_9 = dP$ - Wiener driving force on pitch

$dx_{10} = X_{11}dt + d\theta_1$, θ_1 - Wiener driving force in yaw

$dx_{11} = \frac{x_{12}}{I} dt - \frac{K_2}{I} x_{11}dt$

$dx_{12} = d\tau_1 + d\tau_2$ τ_1 Wiener, τ_2 Poisson driving torques

which is a stochastic differential equation of the form

$$dx = f(x,t)dt + dw$$

This general form for the target portion of the dynamic equation is extended with the case of the clutter and measurement models in the next section to provide the complete stochastic differential equation formulation of the target tracking problem.

SECTION 4. SCENE MODEL

In order to get useful bounds on tracker performance, a scene model which includes the effects of projecting a three-dimensional scene onto a two-dimensional plane is essential. The model must also include the effects of target/clutter occlusion, blurring, spatial sampling, and noise degradations. Any scene model which does not include these effects is unlikely to give useful results.

4.1 PROJECTIONS

Projection effects are most easily handled by using the method of perspective transformations to transform a given point on the object of interest to the image plane. The transformation equations are contained in any book on scene analysis or computer graphics (for example, ref. [39]). Application of these methods to our problem follows:

Let $\vec{r} = [x,y,z]^T$ be the vector coordinates of a point in space relative to a sensor set of fixed coordinates (see figure 4.1). The sensor line of sight axis is assumed to lie along the z axis (third coordinate). Then if the sensor has focal length f, and $f \ll z$:

$$u \cong \frac{x}{z}$$
$$v \cong \frac{y}{z} \qquad (4\text{-}1)$$

where u,v define (in focal length units) the projection of the point on the image plane. Now, if this point is represented as $F' = [x',y',z']^T$ in some other fixed coordinate frame rotated and translated from the sensor's

$$\vec{r} = \underset{\sim}{T}\,\vec{r}' + \vec{r}_o \qquad (4\text{-}2)$$

where $\vec{r}_o$ represents the offset between the two coordinate systems and $\underset{\sim}{T}$

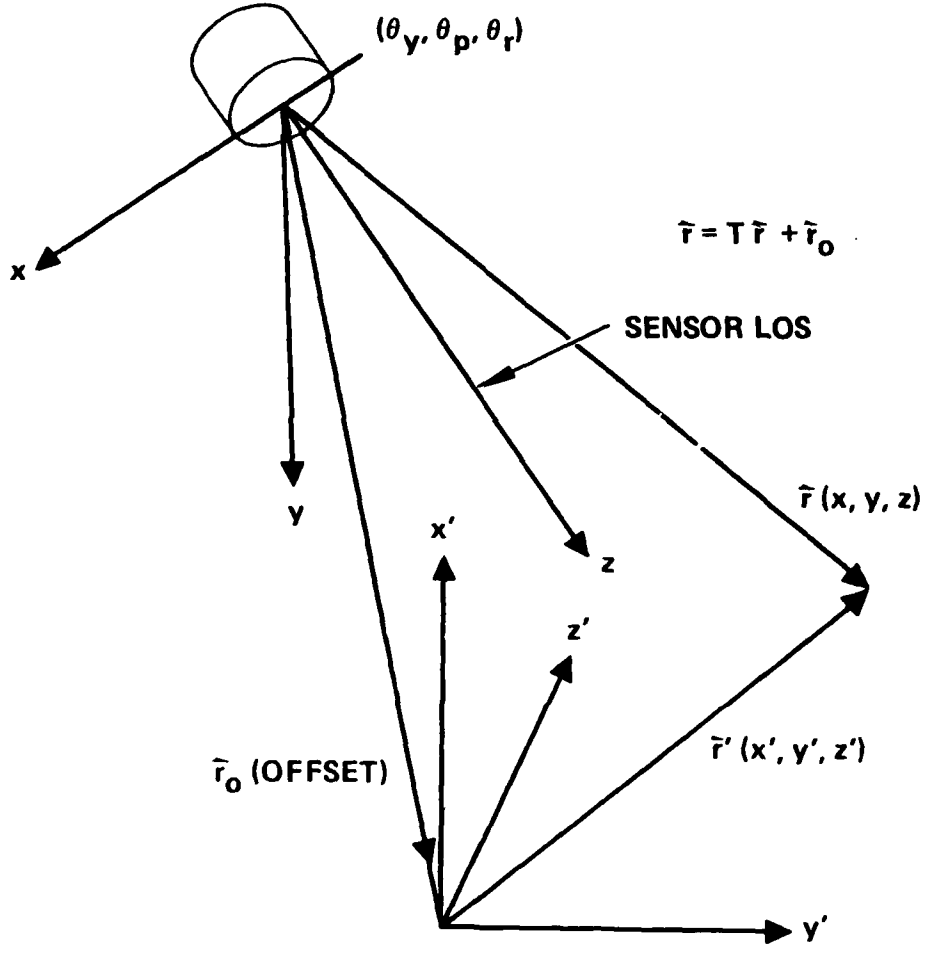


FIGURE 4-1. PROJECTION GEOMETRY

the rotation matrix is a function of $\theta_Y, \theta_P, \theta_R$, the yaw, pitch and roll angles between the two coordinate systems. By substituting (4-2) into (4-1), the projected image coordinates of a given point in space relative to an arbitrary fixed coordinate system can be computed. If an object in physical space has its surface defined by the equation

$$q'(\vec{r}') = 0 \tag{4-3}$$

and on this surface (assumed to be perfrectly Lambertian) it has an emitted I.R. intensity profile $I'(\vec{r}')$, then to predict the sensed image we can again use (4-1) and (4-2) to produce:

$$q(\vec{r}) = q'(\underset{\sim}{T}^{-1}(\vec{r}-\vec{r}_o)) \tag{4-4}$$

$$I(\vec{r}) = I'(\underset{\sim}{T}^{-1}(\vec{r}-\vec{r}_o)) \tag{4-5}$$

where $q(\vec{r})$ and $I(\vec{r})$ are the transformed surface and intensity functions, respectively. Now, representing $q(\vec{r})$ as $q(x,y,z)$, we can produce the function $Q_z(u,v)$ defined by

$$Q_z(u,v) = \begin{cases} \min z \left\{ q(uz,vz,z) = 0 \right\} & \text{if } \exists z \; q(uz,vz,z) = 0 \\ -1 & \text{if } \sim\exists z \; q(uz,vz,z) = 0 \end{cases} \tag{4-6}$$

The interpretation of $Q_z(u,v)$ is that it represents the z axis coordinate of the "closest" part of the object at any given set of projection coordinates subject to the approximation of (4-1). It is negative for a (u,v) position that does not image the target (it is assumed that $q(x,y,z) = 0 \rightarrow z > 0$). Using (4-6) and (4-1), we can solve for the idealized blur and noise-free image of the object, J(u,v) in terms of its transformed intensity profile I(x,y,z).

$$J(u,v) = S(Q) \cdot I(Q\cdot u, Q\cdot v, Q) \tag{4-7}$$

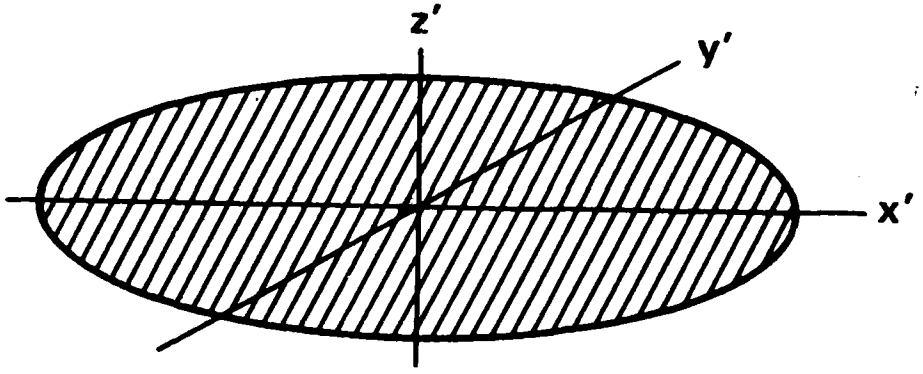
where
$$S(x) = \begin{cases} 0, & X<0 \\ 1, & X\geq 0 \end{cases} \tag{4-8}$$

and $Q = Q_Z(u,v)$.

In practice, J(u,v) is a difficult function to compute analytically. Appendix B treats a related set of transformations for a rectangular solid. Another example could be an ellipsoid as defined in Table 4-1. An ellipse is a good general shape and as shown in the table, can be represented in body coordinates by elliptical equations defined by the three principal axis radii, r_x, r_y, r_z. If the temperature variation over the ellipsoid is negligible, the intensity over the object is constant and the transformation of the object to line-of-sight coordinates can be accomplished as indicated in Table 4-2, where r_o is the distance from the object to the LOS coordinates located at the sensor and T is the yaw-pitch-roll transformation defining the orientation of the object. Since the object has uniform intensity, the 3-dimensional "intensity" in LOS coordinates is just the transformation of each point in the object. The sensed 2-dimensional image can be found by use of a "nearest" function as also shown in Table 4-2. This function provides a means of formalizing the concept that the front of the object blocks out all points in the object behind it. The intensity function J given in the table then is the observed object since it is 1 for front points and zero for points off the object. This concept is extended to include blocking by clutter in the next section.

TABLE 4-1. EXAMPLE OF AN OBJECT MODEL

- OBJECT IN PHYSICAL SPACE



$q'(r') = 0$ r' BODY CENTERED COORDINATE SYSTEM

$I(r')$ – INTENSITY FUNCTION

EXAMPLE – ELLIPSE

$$q'(r') = \frac{(x')^2}{r_x^2} + \frac{(y')^2}{r_y^2} + \frac{(z')^2}{r_z^2} - 1 = 0$$

WHERE $r' = [x', y', z']^T$

$I(r') = \Delta T$

TABLE 4-2. TRANSFORMATION OF SAMPLE OBJECT MODEL TO L.O.S. COORDINATES

$$q(\tilde{r}) = q'(\underset{\sim}{T}^{-1}(\tilde{r} - \tilde{r}_o))$$

$\tilde{r}$ SENSOR CENTERED COORDINATES

$$I(\tilde{r}) = I'(\underset{\sim}{T}^{-1}(\tilde{r} - \tilde{r}_o))$$

DEFINE "NEAREST" FUNCTION

$Q_Z(U, V)$ = z COORDINATE OF CLOSEST POINT TO OBJECT IMAGED AT (U, V)

$$= \begin{cases} \text{MIN } Z \ \{q([UZ, VZ, Z]^T) = 0 \text{ IF } \exists Z \ q([UZ, VZ, Z]^T) = 0 \\ -1 \text{ OTHER} \end{cases}$$

DEFINE INTENSITY FUNCTION

$$J(U, V) = S(Q) \cdot I(QU, QV, Q)$$

$$Q = Q_Z(U, V) \qquad S(x) = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases}$$

4.2 TARGET-CLUTTER OCCLUSIONS

The effects of target and clutter occlusions can now be included. Let Q_{ZT} be the Q function (reference eqn. (4-6)) for the target, and $Q_{ZC}(u,v)$ represent the equivalent Q function for the object composed of the aggregation of all potentially obscuring clutter objects. Similarly, let $J_T(u,v)$ and $J_C(u,v)$ represent the unobscured target and clutter intensity projections, respectively, as in (4-7).

Define the function D(u,v) such that

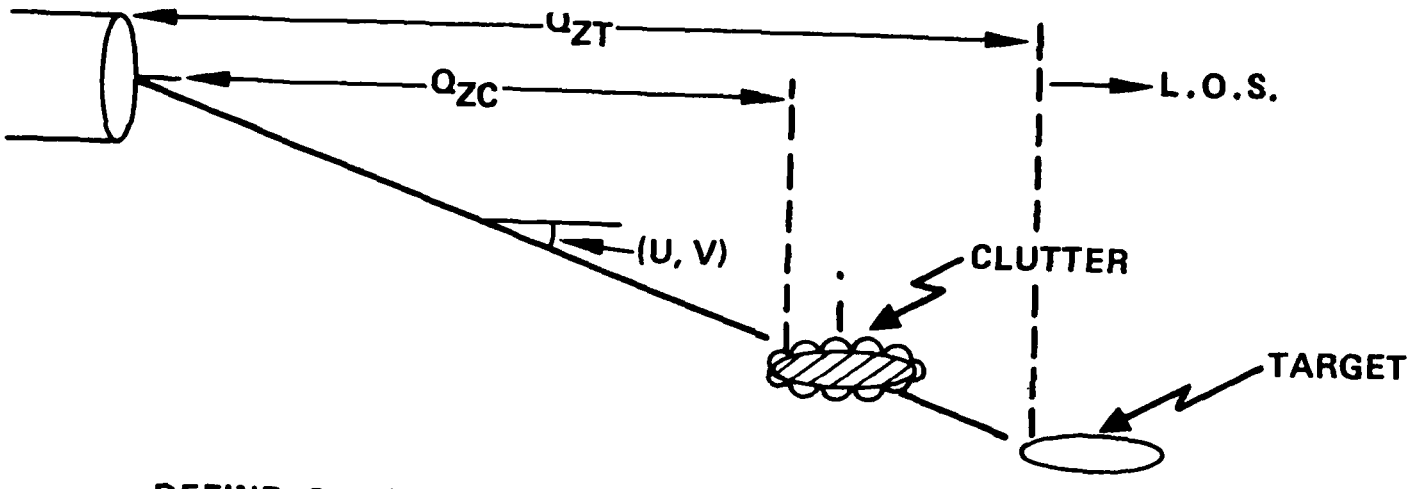
$$D(u,v) = \begin{cases} 1 \text{ if } Q_{ZC}(u,v) < 0 \\ 0 \text{ if } Q_{ZT}(u,v) < 0 \\ S(Q_{ZC}(u,v) - Q_{ZT}(u,v)) \text{ otherwise} \end{cases} \qquad (4\text{-}9)$$

with S(x) as defined in (4-8). Loosely speaking, D(u,v) = 1 when the target obscures the clutter at image position (u,v) and 0 otherwise.

The observed idealized (blur and noise-free) intensity of a target moving in and out of clutter is then given by:

$$J_{TC}(u,v) = D(u,v) \cdot J_T(u,v) + (1-D(u,v))\, J_C(u,v) \qquad (4\text{-}10)$$

as can be seen in Figure 4-2.



DEFINE Q_{ZT} TO BE TARGET Q FUNCTION

Q_{ZC} TO BE CLUTTER Q FUNCTION

DEFINE OCCLUSION FUNCTION

$$D(U,V) = \begin{cases} 1 \text{ IF } (Q_{ZC}(U,V) < 0 \wedge Q_{ZT}(U,V) > 0) \text{ NO CLUTTER} \\ 0 \text{ IF } Q_{ZT}(U,V) < 0 \text{ NO TARGET} \\ S(Q_{ZC} - Q_{ZT}) \text{ OTHERWISE} \end{cases}$$

$$J_{TC} = D \cdot J_T + (1 - D)\, J_C$$

FIGURE 4-2. GEOMETRY OF TARGET CLUTTER

4.3 BLUR AND SAMPLING

First order blurring effects are included by assuming a uniform 2-D blur function. In general

$$h_o(u,v) = \frac{1}{\Delta^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} J_{TC}(\alpha,\beta)\ b_\Delta(u-\alpha,v-\beta)\ d\alpha d\beta \qquad (4\text{-}11)$$

where the blur function $b_\Delta(x,y)$ is given by the system PSF. In this report we will commonly use:

$$b_\Delta(x,y) = \text{rect}(x/\Delta, y/\Delta) = \begin{cases} 1 \text{ if } |x| \le \Delta/2, |y| \le \frac{\Delta}{2} \\ 0 \text{ otherwise} \end{cases} \qquad (4\text{-}12)$$

$h_o(u,v)$ can then be sampled at points (u_i,v_i) to produce the vector observation function $\vec{h}$

$$\vec{h} = \left[h_o(u_1,v_1),\ h_o(u_2,v_2),\ h_o(u_3,v_3)\ldots\right]^T \qquad (4\text{-}13)$$

Sampling intervals can, of course, vary. In many examples in this report we typically use one sample per blur element in a uniform rectangular grid

$$\begin{aligned} u_i &= m(i) \cdot \Delta \\ v_i &= n(i) \cdot \Delta \end{aligned} \qquad m(\cdot), n(\cdot) \text{ integer functions} \qquad (4\text{-}14)$$

4.4 MEASUREMENT MODEL

Once we have selected the $\vec{h}$ function, which has as its implicit variables the state vector describing the target and clutter objects being imaged, noise is added to the model to represent sensor noise in the observation. We will model the noise to be Gaussian, and temporally and spatially white. This is represented by

$$d\vec{y} = \vec{h}(\vec{x})dt + d\vec{z} \qquad (4\text{-}15)$$

where $\vec{x}$ represents the augmented state vector which includes vehicle dynamics, clutter states and/or obscuration variables (states for producing D(u,v) of eqn. (4-9)), and $\vec{y}$ is the observation output for this model.

The covariance of the noise is related to the noise effective temperature of the sensor and frame rate by:

$$E\left[d\vec{z}\ d\vec{z}^T\right] = \frac{(NE\Delta T)^2}{B}\ \underset{\sim}{I}\ dt \qquad (4\text{-}16)$$

where $E[\ ]$ is expectation,

$NE\Delta T$ is the sensor noise effective temperature,

B is the frame rate

$\underset{\sim}{I}$ is the identity matrix.

Note that the output of an actual sensor would be

$$\vec{I}(t) = \int_{t-1/B}^{t} d\vec{y} = \vec{y}(t) - \vec{y}(t-\Delta) \qquad (4\text{-}17)$$

and is not to be confused with $I(\vec{r})$ of equation (4-5). While the equations given by (4-17) are the equations actually satisfied by any real system, the equations of (4-15) and (4-16) are more convenient for mathematical analysis and contain exactly the same statistical information*. Hence, any results obtained when the measurements are specified by (4-15)-(4-16) apply when they are specified by (4-17).

Figure 4.3 illustrates the scene modeling process described in this section.

*In the limit as $B \to \infty$ (fast frame rate).

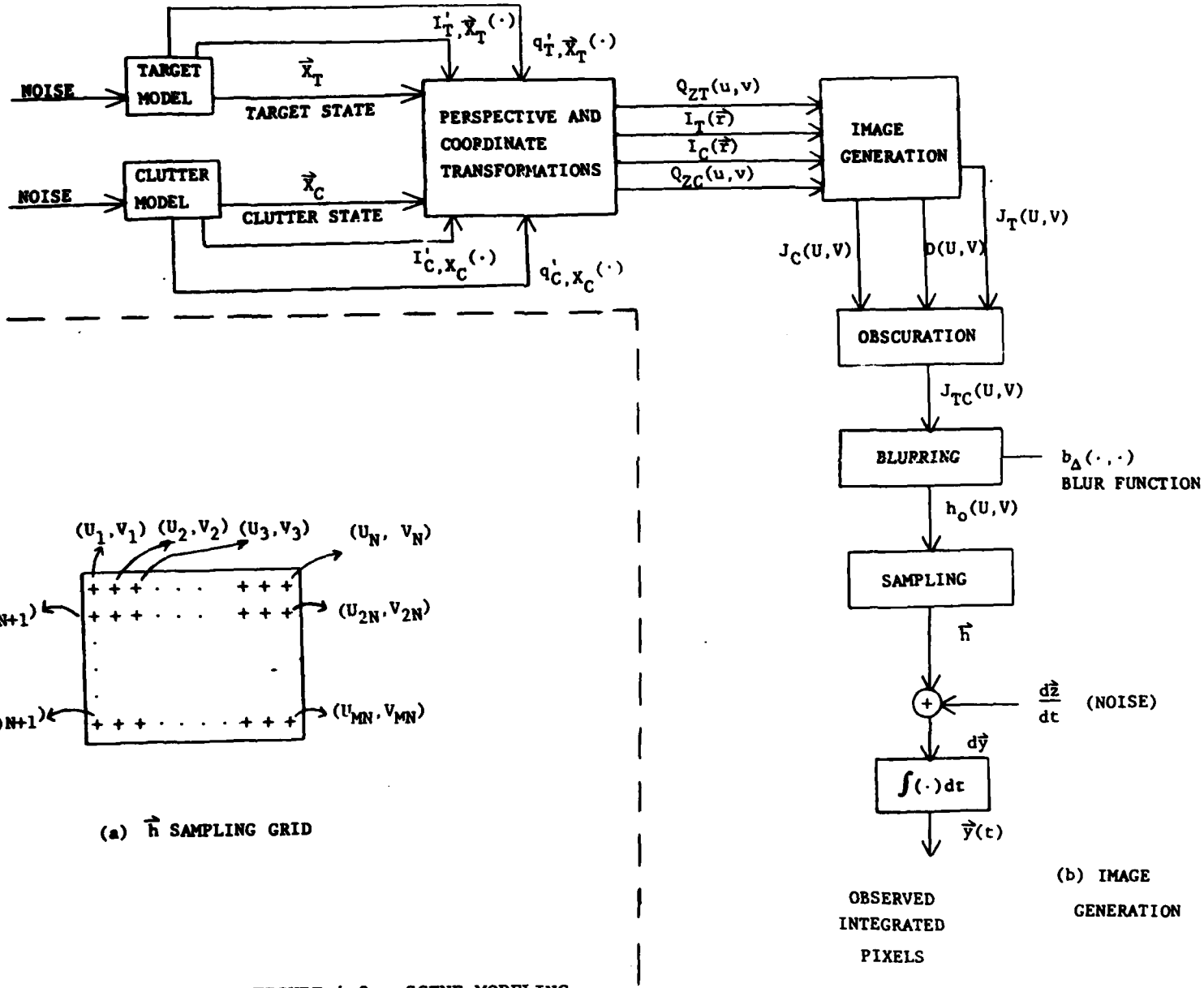


FIGURE 4-3. SCENE MODELING

5.0 ANALYTIC RESULTS

In this section, we introduce two analytical methods for evaluating electro-optical image trackers, namely, linearization, and Bobrovsky-Zakai. Both methods consist of replacing the non-linear tracker model with an equivalent linear model, from which a matrix Riccati equation is derived. A bound or an approximation to the mean square tracking error is then obtained by solving the matrix Riccati equation.

In section 5.1, the Bobrovsky-Zakai method is used to derive a lower bound to the RMS tracking error for a simple rectangular target constrained to move horizontally across the FOV. At any given time, the target could be occluded by a stationary clutter object. The dynamical model used is simple. In section 5.2 and 5.3, the linearization method is used to approximate the RMS tracking error for more realistic dynamics models.

5.1 BOBROVSKY-ZAKAI BOUND

In principle, the Bobrovsky-Zakai method could be applied to the general dynamics-scene model introduced in sections 3 and 4. In section 3, the general dynamics model was given as

$$d\vec{x} = \vec{f}(\vec{x},t)\, dt + C\, d\vec{w}, \qquad (5\text{-}1)$$

where $\vec{x}$ is a 11-dimensional state vector specifing the target's location, velocity, yaw, pitch, roll, and angle rates.

The general scene model given in section 4 conveniently specifies the measurement equation as

$$d\vec{y} = \vec{h}\,(\vec{x}_A)\, dt + D\, d\vec{z} \qquad (5\text{-}2)$$

where $\vec{x}_A$ represents the augmented state vector which includes vehicle

dynamics (the $\vec{x}$ vector in equation 5.1) , and clutter states and/or obscuration variables. $\vec{y}$ is the observation vector. The notation used here is slightly different than that used in section 3 and 4. The notation change is only to facilitate the application of the Bobrovsky-Zakai method. Matrices C and D are related to the forcing function and measurement noise of sections 3 and 4 as follows:

$$CC^T dt = E\{d\vec{w}\, d\vec{w}^T\} \tag{5-3}$$

where $\vec{w}$ is the random forcing function in section 3, and

$$DD^T dt = E\{d\vec{z}\, d\vec{z}^T\} \tag{5-4}$$

where $\vec{z}$ is the measurement noise in section (4).

For the general tracking problem specified by equation (5-1) and (5-2), the Bobrovsky-Zakai bound can be obtained by:

(1) Set $$A_t = E\left\{\frac{\partial \vec{f}(\vec{x})}{\partial \vec{x}}\right\} \tag{5-5}$$

where $\frac{\partial \vec{f}}{\partial \vec{x}}$ is the vector gradient of $\vec{f}$

(2) Solve for B in

$$B^T(DD^T)^{-1}B = E\left\{\left(\frac{\partial \vec{f}}{\partial \vec{x}} - \vec{A}\right)^T (CC^T)^{-1} \cdot \left(\frac{\partial \vec{f}}{\partial \vec{x}} - A\right)\right\} \tag{5-6}$$

$$+ E\left\{\frac{\partial \vec{h}^T}{\partial \vec{x}} (DD^T)^{-1} \frac{\partial \vec{h}}{\partial \vec{x}}\right\}$$

where $\frac{\partial \vec{h}}{\partial \vec{x}}$ is the vector gradient of $\vec{h}$

(3) Replace the nonlinear problem with

$$d\vec{u} = A\,\vec{u}\,dt + C\,d\vec{w} \tag{5-7}$$

$$d\vec{m} = B\,\vec{u}\,dt + D\,d\vec{z} \tag{5-8}$$

(4) Solve the matrix Riccati equation for the linear system specified by equation (5-7) and (5-8), which is given by

$$\frac{dV}{dt} = AV + VA^T + CC^T - VB^TBV \qquad (5\text{-}9)$$

The mean square error V given by the solution to the Riccati equation will then be a lower bound to the mean square error of the nonlinear system (5-1), (5-2).

In general, equations (5-6) and (5-7) cannot be evaluated analytically for a realistic dynamics-scene model; however, both can be easily evaluated by Monte-Carlo methods. The computational burden involved in simulating equations (5-6) and (5-7) will in general be less than that required to simulate the optimal tracker via Kushner's equations or the integral method described in section 6.1.

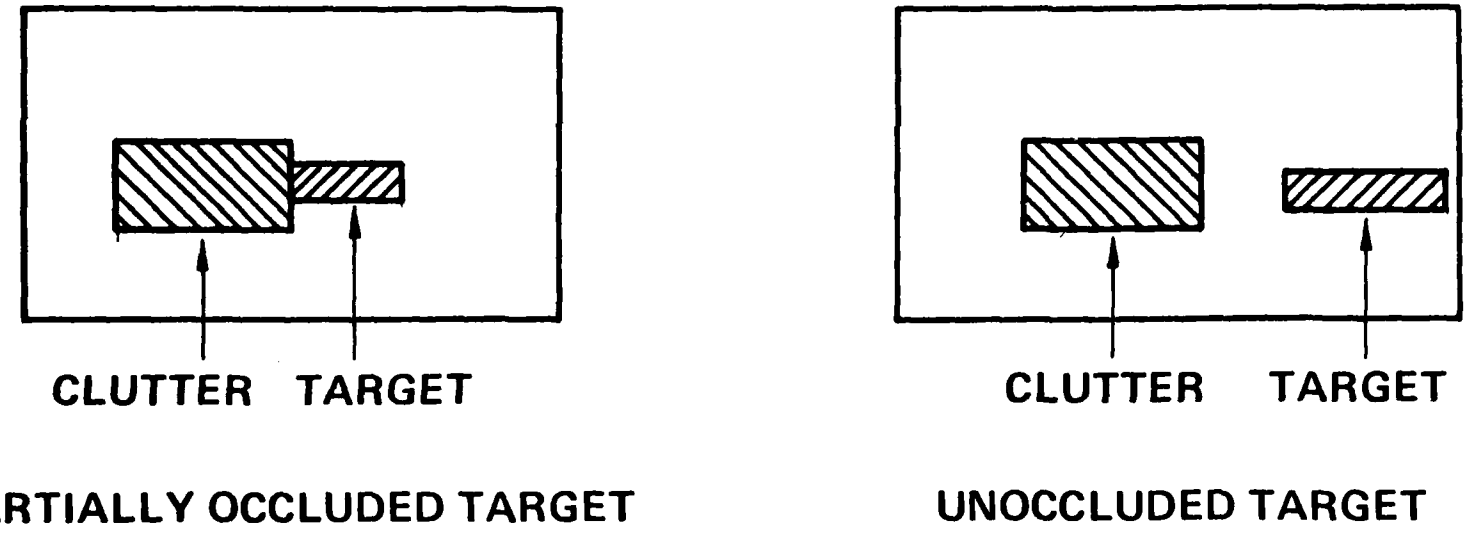
It has been pointed out that for any realistic problem, an easily implemented Monte Carlo simulation will, in general, be required to evaluate certain averages which are used in the Bobrovsky-Zakai bound. However, frequently a qualitative understanding of tracker performance versus system parameters can be obtained by analyzing simple scenarios for which analytic bounds can be obtained.

Hence, the general dynamics-scene model will be restricted to the simple scenario where a rectangular target with fixed size and aspect ratio moves horizontally across the FOV with a Gaussian white-noise velocity. It is further assumed that there is a stationary clutter object in the FOV which at any time might partly or completely occlude the target. (See figure 5.1)
For this simple scenario, equation (5-1) reduces to

$$d\,x_T = \sqrt{q}\,dw \qquad \text{(target motion)} \qquad (5\text{-}10)$$

$$d\,x_C = 0 \qquad \text{(clutter location is fixed)}$$

FIGURE 5.1. SCENE MODEL



where q is the target's velocity spectrum amplitude. The measurement equation (5 -2) reduces to

$$d\vec{y} = \vec{h}(x_T, x_C)\, dt + D\, d\vec{z} \qquad (5\text{-}11)$$

where for pixel i

$$h_i(x_T, x_C) = T_T\, h_{T_i}(x_T)\,(1 - h_{C_i}(x_C)) + T_C\, h_{C_i}(x_C)$$

T_T = target intensity offset

T_C = clutter object intensity offset

$h_{T_i}(x_T)$ = target indicator function

$h_{C_i}(x_C)$ = clutter indicator function

The app'ication of the Bobrovsky-Zakai method results in a RMS error bound given by

$$\text{RMS lower bound} = \left\{ \frac{\sqrt{rq/\alpha_T}\,(1-\exp(-\sqrt{\alpha_T q/r}\; t))}{(1 + \exp(-\sqrt{\alpha_T q/r}\; t))} \right\}^{1/2} \qquad (5\text{-}12)$$

where $$\alpha_T = 2T_T^2\, \frac{N_p}{\Delta^2}\left\{1 - q_{co} + q_{ce/4}\right\}$$

q_{co} = density of clutter pixels = # pixels on clutter object/ # pixels in FOV

q_{ce} = density of vertical clutter edge pixels = # pixels on vertical clutter edge/# pixels in FOV

$r = \frac{(NE\Delta T)^2}{B}$ B = frame rate

T_T = target temperature offset,

q = specturm amplitude of velocity process

Δ = blur size

N_p = number of blur elements across target in the vertical direction.

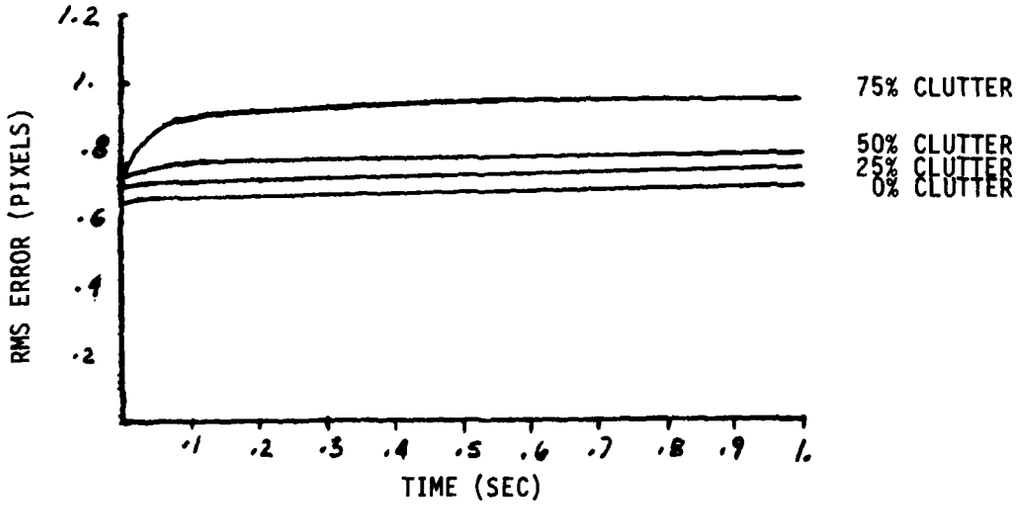
A detailed derivation of this result is given in appendix E.

Figure 5.2 demonstrates the variation of this bound as a function of fraction of effective FOV covered by the clutter object, where effective FOV is defined as that part of the FOV for which it is possible for the target to lie.

EFFECTS OF CLUTTER ON IMAGE TRACKING

FIGURE 5.2

- o SNR = $\Delta T/NE\Delta T$ = 1
- o B = 30HZ
- o q = 3 $(PIXELS/SEC.)^2/HZ$
- o TARGET SIZE = 5 x 5 PIXELS
- o TARGET AND CLUTTER POSITION ASSUMED KNOWN AT TIME 0.



5.2 APPLICATION OF LINEARIZATION TO I-D FORCE MODEL

A second approach to performance analysis of imaging trackers is to find approximate results by linearization. Starting with the simplest dynamics model these linear equations are solved to provide analytic relationships between the RMS tracker error and the appropriate descriptive parameters. The results are extended to progressively more general dynamics models to find generalizations which relate desired figures of merit (e.g. RMS error) to system parameters.

Use of linearization results in an extended Kalman filter, which can then be solved to get an approximate solution. This section shows how that approach can lead to useful functional relationships and point the way to generalizations of particular interest in the error covariance matrix V, which, after linearization, satisfies a Riccati equation of the form:

$$\frac{dV}{dt} = AV + V A^T + M - VNV \qquad (5\text{-}13)$$

where A is the dynamics matrix, M is a stochastic disturbance matrix and N is a measurement matrix.

Three cases of the equation for a linearized Kalman filter with one-dimensional motion have been analyzed, and closed-form solutions have been obtained for each case. In each of these cases, the dynamics equation becomes linear, so that the linearization process need be applied only to the measurement equation.

The one-by-one case is the Brownian motion and the target position is regarded as a one-dimensional process.

The second case is a two-state problem (two-by-two matrices) where the acceleration is white noise, so that the motion is affected by an integrated process.

The third case is a three-state problem (three-by-three matrices) where the jerk (i.e., the time derivative of acceleration) is a white noise process. Thus, the motion is affected by a doubly integrated process.

The velocity, acceleration, and jerk spectral densities for the three cases, respectively, Q_1, Q_2, and Q_3, are defined in Table 5-2, and define the M matrix for each case. Note how the time dependence changes for each case, and that the reciprocal of the observation jitter spectral density, q^{-1}, appears as the upper left element in each of the N matrices.

The jitter spectral density, q, may be shown to be, in effect, the same quantity developed in the ATAC Autocuer Modeling Study report [30] using the Cramer-Rao bound to estimate the uncertainty in estimating the position of a target. This can be seen as follows: The quantity q is a product of three factors, of which the first, $(\delta/SNR)^2$, is the variance of a single edge pixel. The second factor, δ/W, the reciprocal of the number of pixels on a target edge, gives the spatial integration effect in reducing the edge variance. The third factor, 1/B, relates the jitter amplitude to a jitter spectral density.

The solutions to the equation for the three cases are given in Table 5-3. Note that for the steady-state case, the observation spectral density q appears as a linear factor in each case multiplied by the reciprocal of a certain time constant. In fact, the steady state for all three cases is given by the formula

$$V_{11}(\infty) = q \cdot T_n^{-1} \qquad (5\text{-}14)$$

where

$$T_n = 2^{-(n-1)/2} (q/Q_n)^{1/(2n)}$$

for n = 1,2,3, respectively.

TABLE 5-1. RICCATI EQUATION APPLIED TO VARIOUS MODELS

1 x 1 MATRIX — BROWNIAN MOTION
2 x 2 MATRIX — WHITE NOISE FORCE
3 x 3 MATRIX — GENERALIZED POISSON PROCESS FORCE

TABLE 5-2. DEFINITIONS OF MATRICES APPEARING IN RICCATI EQUATION FOR LINEARIZATION EXAMPLES

1×1 Case $A = 0 \quad M = Q_1 \quad N = q^{-1}$

2×2 Case $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad M = \begin{pmatrix} 0 & 0 \\ 0 & Q_2 \end{pmatrix} \quad N = \begin{pmatrix} q^{-1} & 0 \\ 0 & 0 \end{pmatrix}$

3×3 Case $A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & Q_3 \end{pmatrix} \quad N = \begin{pmatrix} q^{-1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

q — Observation Jitter Spectral Density $\left(\frac{\text{mrad}^2}{\text{Hz}}\right)$

Q_1 — Velocity Spectral Density $\left(\frac{\text{mrad}}{\text{sec}}\right)^2/\text{Hz}$

Q_2 — Acceleration Spectral Density $\left(\frac{\text{mrad}}{\text{sec}^2}\right)^2/\text{Hz}$

Q_3 — Jerk Spectral Density $\left(\frac{\text{mrad}}{\text{sec}^3}\right)^2/\text{Hz}$

$q = \left(\frac{\delta}{\text{SNR}}\right)^2 \left(\frac{\delta}{W}\right)\left(\frac{1}{B}\right) \quad /2$

TABLE 5-3. RESULTS OF LINEARIZATION EXAMPLE

Brownian Motion Variance:

$$V_{11}(\infty) = q/T_1^{-1}$$

$$T_1 = \left[\left(\frac{\delta}{\sqrt{Q_1 B}}\right) \cdot \left(2 \cdot SNR^2 \left(\frac{W}{\delta}\right)\right)^{-1/2}\right]$$

White Noise Force Variance:

$$V_{11}(\infty) = L_{11}^{\ 2} = q\ T_2^{-1}$$

$$T_2 = \left[\frac{\delta^{1/2}}{(Q_2\ B)^{1/4}} \cdot \left(8(SNR)^2 \left(\frac{W}{\delta}\right)\right)^{-1/4}\right]$$

White Noise Jerk Variance:

$$V_{11}(\infty) = q\ T_3^{-1}$$

$$T_3 = \left[\frac{\delta^{1/3}}{(Q_3\ B)^{1/6}} \left(2^7\ (SNR)^2 \left(\frac{W}{\delta}\right)\right)^{-1/6}\right]$$

Seconds — Signal to Noise

Details of the development of the solution of second and third cases are given in Appendices C and D. Here, the procedure is applied to the first case to illustrate this method. In this case, the equations reduce to two scalar equations:

$$dx = dw \tag{5-15}$$

$$dy = h(x)\,dt + dz \tag{5-16}$$

where dw and dz are Wiener Processes.

The linearization applied to the measurement is the scalar degenerate version of the linearization given in Appendix C and the variance V obeys the equation.

$$\frac{dv}{dt} = M - V N V = Q_1 - \frac{1}{q} V^2 \tag{5-17}$$

Where Q_1 and q are as defined in Table 5-2. The solution of this scalar differential equation from an initial condition $V = 0$ is

$$V = a \tanh bt \tag{5-18}$$

Direct substitution of this solution into the differential equation provides a solution for the constants a and b as:

$$a = \sqrt{Q_1 q} \tag{5-19}$$

$$b = \sqrt{Q_1/q} \tag{5-20}$$

so that

$$V = \sqrt{Q_1 q}\, \tanh \sqrt{Q_1/q}\, t \tag{5-21}$$

5.3 APPLICATION OF LINEARIZATION TO TWO-DIMENSIONAL FORCE MODEL

This model is like the one-dimensional model, except that the target is permitted to move in two perpendicular directions with random white-noise force components in both directions. This is therefore a 4-state problem in which the state variables are defined in terms of the x and y components and velocities as follows:

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} \\ x_3 &= y \\ x_4 &= \dot{y} \end{aligned} \qquad (5\text{-}22)$$

Thus the covariance matrix has sixteen (equals four times four) components, of which six are the same. Thus one is led to a Riccati equation equivalent to ten coupled non-linear differential equations.

When one imposes the requirement that the covariance matrix elements are all to be zero initially at time zero, one finds upon examining the equations that the matrix elements correlating x and y positions and velocities, namely V_{13}, V_{14}, V_{23}, and V_{24}, will remain zero for all time, since they have no non-zero forcing terms. Thus the x and y equations decouple into two sets of three equations each. Each set is equivalent to the set of coupled non-linear differential equations solved in the preceding section. Thus the solutions are essentially the same, except for x and y parameters which may be different for the two directions.

For example, the steady state tracking error for the x and y directions is given by

$$\sigma_x = (4\ Q_x/q_x^3)^{1/8} \tag{5-23}$$

$$\sigma_y = (4\ Q_y/q_y^3)^{1/8} \tag{5-24}$$

where $q_x = \frac{2W}{\delta^3}\ (SNR)^2\ B$ (5-25)

as before with W the target width, but

$$q_y = \frac{2L}{\delta^3}\ (SNR)^2\ B \tag{5-26}$$

with L the target length. Even if Q_x and Q_y are equal, the tracking error for the two directions will be different if the target's length and width are unequal.

SECTION 6. SIMULATION

6.1 Introduction

The purpose of the simulation is two-fold. First, it provides direct verification of the validity of some of the calculations performed in the preceding section. Secondly, it provides a means for directly extracting characteristics of the probability density of more complicated systems, whose performance may be more difficult to estimate in a tractable fashion.

6.2 Approach

The approach that was employed in the computer simulation is to first approximate the continuous time differential models with appropriate discrete time difference models, i.e.,

$$\begin{aligned} \vec{x}(K) - \vec{x}(K-1) &= \vec{g}(\vec{x}(K-1)) + \vec{W}_1(K-1) \\ \vec{y}(K) - \vec{y}(K-1) &= \vec{h}(\vec{x}(K-1)) + \vec{W}_2(K-1) \end{aligned} \qquad (6\text{-}1)$$

or equivalently letting

$$\vec{Z}(K) = \vec{y}(K) - \vec{y}(K-1)$$

$$\vec{f}(\vec{x}(K-1)) = \vec{g}(\vec{x}(K-1)) + \vec{x}(K-1)$$

we have from (6-1)

$$\begin{aligned} \vec{X}(K) &= \vec{f}(\vec{x}(K-1)) + \vec{W}_1(K-1) \\ \vec{Z}(K) &= \vec{h}(\vec{x}(K-1)) + \vec{W}_2(K-1) \end{aligned} \qquad (6\text{-}2)$$

where $\vec{x}(k), \vec{z}(k)$ are intended to represent the state and differenced observation vectors respectively at the k^{th} time epoch, i.e. $t = k\Delta t$. $\vec{w}_1(k)$ and $\vec{w}_2(k)$ are multivariate independent (each dimension and for each k) Gaussian random variables.

Under this set of assumptions $J_k(\vec{u})$, the probability density of the state vector at the k^{th} interval, given observations of $\vec{z}(i)$ for $i \leq k$ is given by (ref 4):

$$P_{\vec{x}(K)|\vec{z}}(\vec{u}|\vec{z}(i),\ i \leq K) \equiv J_K(\vec{u})$$

$$J_K(\vec{u}) = \frac{L_K(\vec{u})}{\int L_K(\vec{u})\, d\vec{u}} \tag{6-3}$$

where $L_k(\vec{u})$ (an unnormalized probability density) is given by

$$L_K(\vec{u}) = N(\vec{z}(K) - \vec{h}(\vec{u}), \underset{\sim}{R}) \times \int N((\vec{u} - \vec{f}(\vec{v})), \underset{\sim}{Q})\ J_{K-1}(\vec{v})\, d\vec{v} \tag{6-4}$$

where

$$\underset{\sim}{R} = E[\vec{w}_2(K)\, \vec{w}_2(K)]$$
$$\underset{\sim}{Q} = E[\vec{w}_1(k)\, \vec{w}_1(K)] \tag{6-5}$$

are the noise covariance matrices and $N(\vec{A}, \underset{\sim}{B})$ is proportional to the Gaussian multivariate density

$$N(\vec{A}, \underset{\sim}{B}) = \exp\left[-\frac{1}{2}\,(\vec{A}^T\, \underset{\sim}{B}^{-1}\, \vec{A})\right] \tag{6-7}$$

Figure 6.1 illustrates the operation of the simulation. In the first stage, prior to actual execution of the simulation, subroutines representing the $\vec{f}$ and $\vec{h}$ functions are linked in the remainder of the simulation to produce

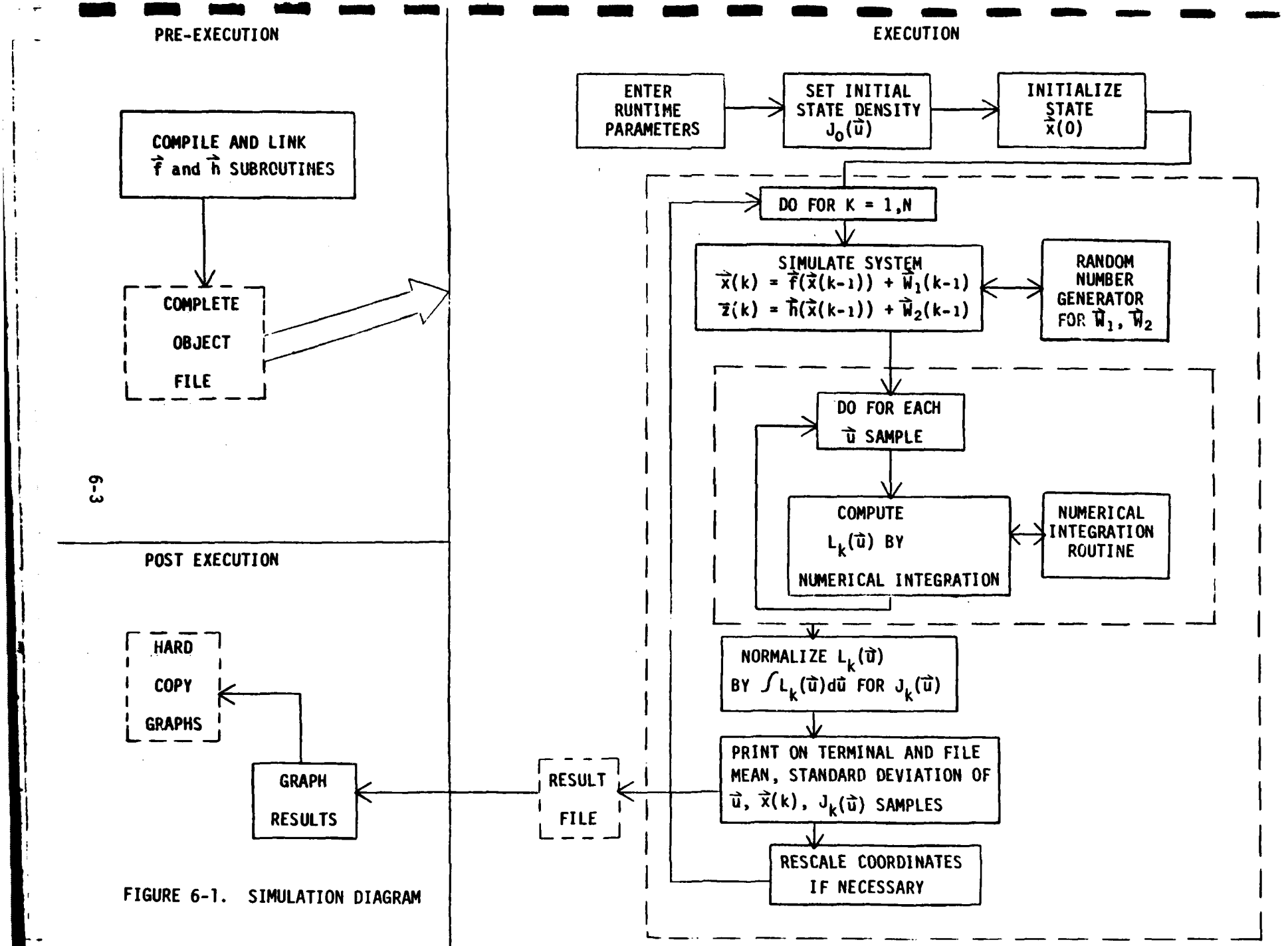


FIGURE 6-1. SIMULATION DIAGRAM

an executable object file. Following this stage, execution begins with the routine parameters being entered from the keyboard. These include output modes, number of iterations (N), dimensionality of state and observation vectors, $\underset{\sim}{Q}^{-1}$,$\underset{\sim}{R}^{-1}$ diagonal elements (since these are assumed diagonal matrices), number of samples of $\vec{u}$ per dimension, starting state for the system, and initial state probability density parameters.

After this $J_o(\vec{u})$, which is represented by values on a uniform sampling grid, is initialized as is the initial system state.

Then for each iteration of the simulation, the system producing observations is simulated via the $\vec{f}$ and $\vec{h}$ subroutines and a random number generator to produce simulated noise. Following this, the next $L_k(\vec{u})$ function is again computed, point by point, on the same sampling grid that $J_{k-1}(\vec{u})$ was defined on, using numerical integration techniques. It is in this stage that most of the simulation's throughput is used. After $L_i(\vec{u})$ is computed, it is normalized to produce the new $J_k(\vec{u})$ and the results are printed out on the terminal and a file. The sampling grid is then redefined to accommodate the new probability density (allowing for spreading and shifts) and the next iteration begins.

Following the execution stage, a graphing program can read the files containing the results and produce graphs of time tracks of estimated means, standard deviations, or probability densities to allow more readily interpretable results.

6.3 Simulation Comparison

Due to the stochastic nature of the simulation, any quantity observed during a single run of the simulation should be considered as the outcome of a random experiment and combined with the results of other simulation runs to obtain an estimate of the quantity of interest. Here the quantity of interest is RMS tracking error, and it is estimated from the output of the simulation as follows:

$$\widehat{RMS}(t_i) = \left(\sum_{k=1}^{N} \left(e_k(t_i) - \bar{e}(t_i) \right)^2 / (N-1) \right)^{\frac{1}{2}}$$

where $e_k(t_i) = X_k(t_i) - \hat{X}_k(t_i)$, $\bar{e}(t_i) = \frac{1}{N} \sum_{k=1}^{N} e_k(t_i)$,

N is the number of simulation runs, $X_k(t_i)$ is the target's location at time t_i for the k^{th} simulation run, and $\hat{X}_k(t_i)$ is the corresponding optimal estimate of $X_k(t_i)$.

In figure 6.2, both $\widehat{RMS}(t_i)$ and the Bobrovsky-Zakai bound are given for a 5 x 5 pixel^2 target which is moving horizontally across the FOV with a Gaussian white noise velocity. Although the target is square these results hold equally well for rectangular targets since the motion is one dimensional. The height of the target is just used to build up SNR. The target's velocity spectrum amplitude is assumed to be 3 (pixel/sec)^2/Hz and SNR = 1. Also, the α_T of equation (5-12) would be one, since there are no clutter objects in the scene.

In figure 6.3, $\widehat{RMS}(t_i)$ and the RMS error obtained by the linear approximation method are given for a 2 x 2 pixel^2 target which moves horizontally across the FOV, satisfying the dynamics equation

$$dX = V\,dt$$

$$dV = \sqrt{q}\,dw$$

where X is the target location and V is its velocity. SNR is assumed to be 1 and the spectrum amplitude of the target's acceleration is 1.9 (pixels/sec^2)^2/Hz. For this simple example, the results from the linear approximation agree well with the simulation.

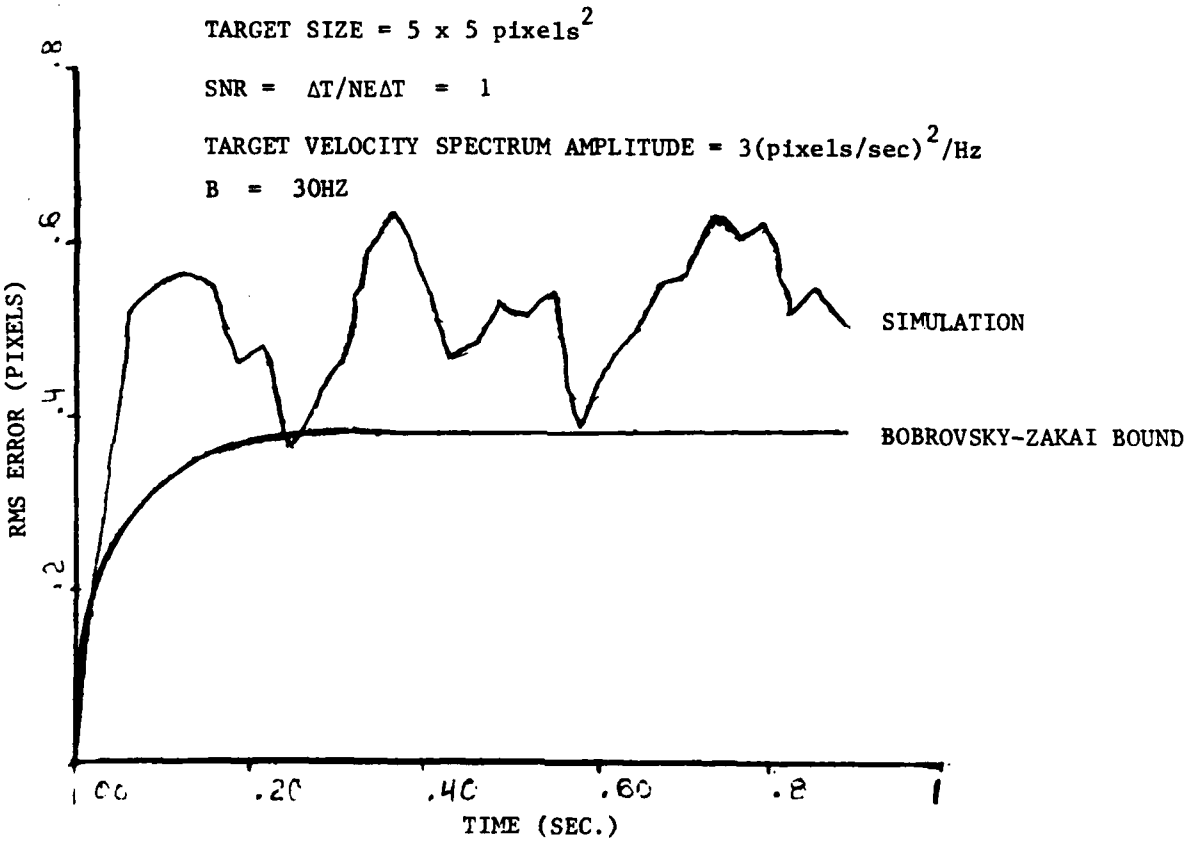


FIGURE 6-2. BOBROVSKY-ZAKAI BOUND

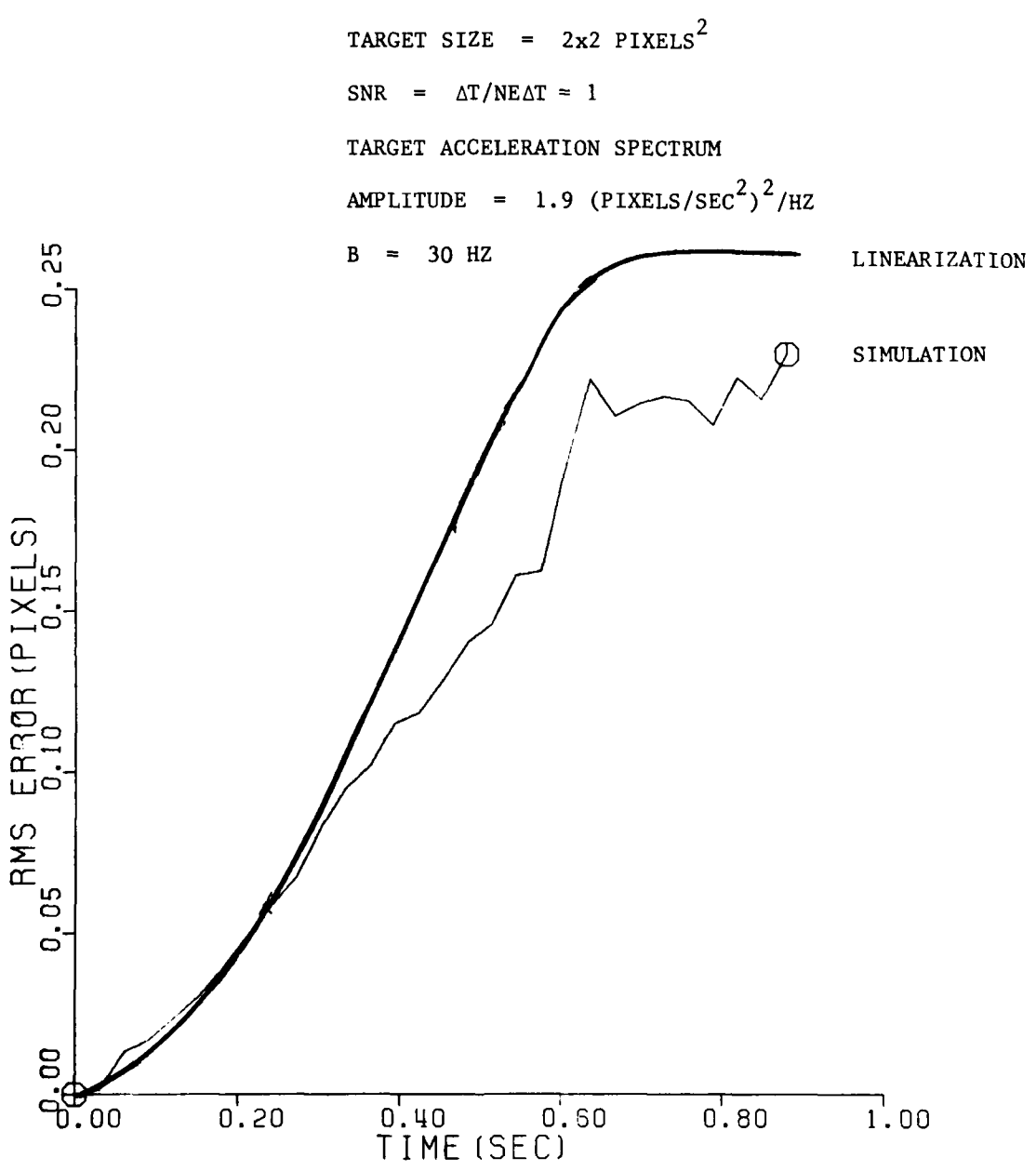


FIGURE 6-3. LINEARIZATION METHOD

In order to run the discrete simulation, the following inputs are required:

1). $R = NE\Delta T^2$

2). $Q = q/\Delta f$

where, for the B-Z example q is the velocity spectrum amplitude and Δf is the frame rate. For the linearization example q is the acceleration spectral amplitude. The values of all parameters necessary to run the simulation for the B-Z example and the linearization example are given in tables 6-1 and 6-2, respectively.

TABLE 6-1. INPUTS FOR B-Z EXAMPLE

# OF TRIALS	30
TARGET SIZE	5 x 5 pixels^2
FOV	25 x 5 Pixels^2
DELTA T	1.
NUMBER OF STATES	1
INVERSE Q	$(30)^{-1}$
INVERSE R	1
INITIAL TARGET STATE	ASSUMED KNOWN
NUMBER OF TIME SAMPLES	30

TABLE 6-2. INPUTS FOR LINEARIZATION EXAMPLE

# OF TRIALS	10
TARGET SIZE	2 x 2 pixels^2
FOV	10 x 2 pixels^2
DELTA-T	1
NUMBER OF STATES	2
INVERSE Q	$(1.9/30)^{-1}$
INVERSE R	1
INITIAL TARGET STATE	ASSUMED KNOWN
NUMBER OF TIME SAMPLES	30

SECTION 7. APPLICATION TO PRECISION AIMPOINT MAINTENANCE

The analytical methods and results descirbed in this report have application in the design and analysis of electro-optical systems requiring tracking functions to be performed. A key applications area includes precision aimpoint maintenance systems.

Precision aimpoint maintenance is a required function for systems such as directed energy weapons. In this application, low tracker jitter and RMS error from aimpoint is a requirement. Since small deviations from the target position are relevant in this application the linearization approach outline in section 5.3 will be used in the following example.

7.1 Target Model

The target to be tracked in this case will be modeled to be imaged as a rectangle of length L (radians)*, width W (radians), and intensity offset ΔT ($^{\circ}C$), blurred by a characteristic blurring function of dimension r (radians). In other words:

$$I_{u,v}(x,y) = \Delta T \cdot \text{rect}\left(\frac{x-u}{L}, \frac{y-v}{W}\right) \otimes \text{rect}\left(\frac{x}{r}, \frac{y}{r}\right) \tag{7-1}$$

where

$$\text{rect}\,(x,y) = \begin{cases} 1, & |x| \leq 1/2,\ |y| \leq 1/2 \\ 0, & |x| > 1/2 \text{ or } |y| > 1/2 \end{cases}$$

and $\otimes$ is convolution

$I_{u,v}(x,y)$ represents the perceived intensity of a target located at image position (u, v) detected at sensor azimuth x and elevation y. The motion of the target (relative to the sensor line of sight) will be modeled by second order linear stochastic differential equations in the azimuth and elevation directions

$$\begin{aligned} du &= \dot{u}\, dt \\ du &= d\, w_u \\ dv &= \dot{v}\, dt \\ dv &= d\, w_v \end{aligned} \tag{7-2}$$

with w_u, w_v being uncorrelated Wiener processes.

7.2 Sensor Model

The sensor model will treat the sensor as providing continuous samples in time of a discretely sampled (in space) intensity function. The sampling

* Using radians is equivalent to using focal length units of section 4

interval is r radians, with h (odd) samples subtending the azimuth and elevation FOV. Each sample will be corrupted by an independent white noise source, i.e.

$$d\vec{J} = \vec{\Phi}\, dt = d\vec{Z} \tag{7-3}$$

where $\vec{J}$ is a vector of length n^2 representing the observed intensity samples

$\vec{Z}$ is a vector Wiener process

$$E[d\vec{Z}\, d\vec{Z}^T] = T_n\, I\, dt$$

where T_n is $(NE\Delta T)^2 / B$, B = frame rate

$\vec{\Phi}$ is a vector representing the n^2 samples on the image plane

$$\vec{\Phi} = [\phi_{11}, \phi_{2,i}, \ldots, \phi_{m1}, \phi_{12}, \phi_{22} \cdots \phi_{mm}]^T \tag{7-4}$$

where

$$\phi_{ij} = f(i' \cdot r/2,\ j' \cdot r/2)$$

$$i' = i - (m-1)/2,\quad j' = j - (m-1)/2$$

and $f(x,y)$ is the focal plane intensity (^oC) at azimuth x, elevation y.

Sensor motion will be accounted for by equation (7-2) since the angular motion of the target is defined relative to the sensor line of sight.

7.3 Combined Model

Combining the sensor and target models

$$du = \dot{u}\, dt \qquad dv = \dot{v}\, dt$$

$$d\dot{u} = d\, w_u \qquad d\dot{v} = d\, w_v$$

$$d\vec{J} = \vec{\Phi}(u,v)\, dt + d\vec{Z}$$

$$\phi_{ij}(u,v) = I_{u,v}\left(\left[i - \frac{(n-1)}{2}\right] r/2,\ \left[j - \frac{(n-1)}{2}\right] r/2\right) \tag{7-5}$$

$$I_{u,v}(x,y) = \text{rect}\left(\frac{x-u}{L}, \frac{y-v}{w}\right) \circledast \text{rect}\,(x/r, y/r)$$

(7-5) is an example of a linear system - non linear observation

7.4 Performance Analysis

As previously stated, this application is concerned with a low jitter and RMS error from aimpoint requirement. Thus, the results of the linearization method introduced in section 5.3 will be used.

The steady state tracking error (from section 5.3) is

$$\sigma_{EL} = (4Q_{EL}/q_{EL}^3)^{1/8} \quad \text{for the elevation direction} \qquad (7\text{-}6)$$

and

$$\sigma_{AZ} = (4\,Q_{AZ}/q_{AZ}^3)^{1/8} \quad \text{for the azimuth direction} \qquad (7\text{-}7)$$

where

Q_{AZ}, Q_{EL} = spectrum amplitude of the angular acceleration due to jitter

$$q_{EL} = \frac{2W}{\delta^3}(SNR)^2_B$$

$$q_{AZ} = \frac{2L}{\delta^3}(SNR)^2\,B$$

B = frame rate

δ = blur size.

In order to preceed further, the following sensor parameters are assumed

δ = blur size = 180 μrad

B = frame rate = 30 HZ

Q_{AZ}, Q_{EL} = 18.7 $(mrad/sec^2)^2$/HZ

The target size is assumed to be

W = .66 mrad

L = 1.9 mrad

The elevation and azimuth steady state RMS tracking error for these parameters are given in figure 7.1. This figure could then be used to set performance requirements. For example, if an aimpoint accuracy of 30μ rad is required, then it is seen that SNR = $\Delta T/NE\,\Delta T$ must be no lower than 2.5.

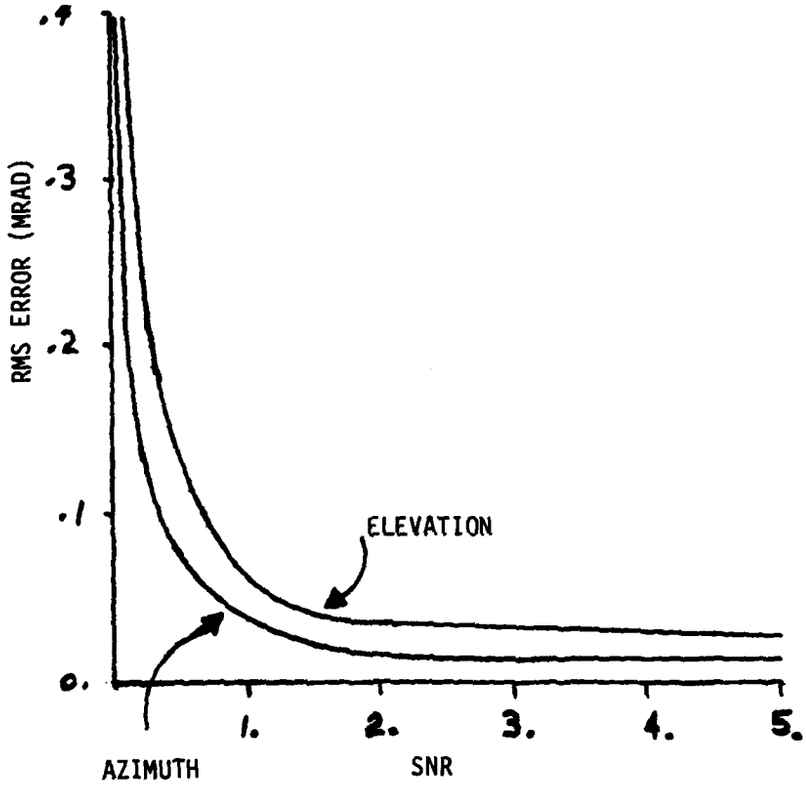


FIGURE 7-1. AIMPOINT ACCURACY

SECTION 8. CONCLUSIONS

This section will summarize the work performed under this contract and draw certain conclusions related to this work.

The first conclusion one can draw based on the study of both the target and the scene model, as well as the definition of the tracker, is that the technique of applying non-linear stochastic differential equations has the power to completely model electro-optical tracking systems. A key point to be noted here is that both the target dynamics as well as sensor noise can be described using the same types of Martingale processes and can be treated in a mathematically unified manner. The model that has been developed has the ability to include maneuvers, clutter, target obscuration, and can include sensor blur, noise, and sampling effects as well. This ability to treat the general problem in a unified way, with a given set of noises driving both the state and observation vectors, has a great deal of descriptive power as well as mathematical elegance.

The second point to note is that for this particular type of model an exact state density solution exists. It exists in a number of forms, one of which is the Kushner equation and the second is the representation theorem. Since this solution exists, one can in theory solve for an optimal tracker under the conditions of the model and this optimal tracker is completely mathematically well defined.

All necessary bounding properties of a given tracker can be extracted from looking at the properties of the solution. The solution is, however, computationally expensive, and statistical as well. Monte Carlo runs may be required to find the average properties of the solution and Monte Carlo runs of the computationally intensive solution can be prohibitively expensive.

By studying the exact state density solution one can get insight into the proper ways in which one can design a tracker. As an example of this, an intelligent

tracker design recently proposed by Hughes was motivated from a study of the exact solution generated by the Kushner equation.

Since solving the Kushner equation in real time is impractical with the current state of technology, and even solving it off-line as a method of analysis can be impractical for a tracker containing a large number of states, the use of analytic techniques to bound and approximate the properties of the solution has been examined. Among these are: the Bobrovsky-Zakai method which is basically an extension of previous work on providing a Cramer-Rao type bound on the performance of a system; the technique of cone bounded non-linearities which can bound tracker performance relating to the degree of non-linearity of the system; extended Kalman techniques which can provide a very precise indication of the accuracy one would obtain for small variations in target state; and methods of analyzing non-linear stability in terms of calculating the time for a given tracker to completely lose track of the target.

At this point, it is seen that several extensions and improvements to these given techniques would be very desirable. Tightening the existing bounds, experimenting with various Liaponov functions for computing stability, and extending the Bobrovsky-Zakai bound to handle Poisson inputs are three areas that could benefit from more study.

Once one has the analytic techniques, in principle one can estimate how close to optimal any given sub-optimal design will fall, and one can derive the functional relationships between the system parameters which would be of particular utility in the specification of trackers.

These bounds and approximations that could be useful for estimating and specifying the performance of the tracker have been verified directly by simulation. In the process several solutions to Kushner's equation have been generated. Applications have been explored, specific cases of tracking N-state random target motion have been studied with respect to jitter properties and a procedure by which one can describe the general and formal procedure for defining and designing a tracker has been initiated.

Appendix A - Application of MACSYMA

MACSYMA is a powerful system of computer programs, developed at MIT, for performing computer algebra. (An interesting article on computer algebra appears in the December, 1981 Scientific American.) MACSYMA has the capability of symbolically performing algebra, expansion and distribution of terms, combining like terms, factoring and other types of simplification, calculus including differentiation and integration, solution of differential equations, matrix calculus including multiplication of matrices and finding inverses, and more.

MACSYMA has been used in the past to perform calculations that either have or would have required months or years to perform by hand. For example, computer scientists, in about 20 hours of computer time, have obtained formulas which reproduce (and correct) formulas first obtained by the 18th century astronomer, Charles Delaunay, which he published in two volumes after twenty years of derivation, checking, and rechecking. Surprisingly, only three small errors in Delaunay's work were found. Further work has modeled effects not treated in Delaunay's work.

This points out the main use of MACSYMA for either greatly reducing the time required to derive and check a difficult result, or in obtaining results which one could not obtain at all by hand.

---

*This work was done with the aid of MACSYMA, a large symbolic manipulation program developed at the MIT Laboratory for Computer Science and Support by the National Aeronautics and Space Administration under Grant NSG 1323, by the Office of Naval Research under Grant N00014-77-C-0641, by the U.S. Department of Energy under Grant ET-78-C-02-4687, and by the U.S. Air Force under Grant F49620-79-C-020. Special thanks are due Leo P. Harten for supplying valuable information on use of the MACSYMA system, and for helping us overcome problems we have encountered in using it.

Our Use of MACSYMA

Because we were continually running into problems involving lengthy, tedious algebra, we decided to use MACSYMA. Our initial use of MACSYMA was to perform algebra involved in calculating bounds formulas for translational and rotational target models. Although rigorous formulas were obtained, the bounds proved to be too small in practice to be of practical use.

Consequently, an approach based upon linearizing the measurement model was undertaken. This led to matrix Riccati equations for the various models: A 2x2 matrix Riccati equation for the one-dimensional dynamic model with white noise driving force; a 3x3 matrix Riccati equation for the same model with Poisson type force; a 4x4 matrix Riccati equation and a 6x6 matrix Riccati equation, which generalize the previous two models to permit y-motion as well as x-motion.

Even the simplest of these would be extremely difficult to do by hand, involving tedious checking and rechecking. However, with the use of MACSYMA, closed form expressions have been obtained for all of these problems and verified, an extremely difficult task to do by hand, since an explosion of terms occurs before final cancellation.

MACSYMA has also been employed to derive the form of the differential equations which must be used to obtain the Rhodes-Gilman cone-bounded non-linearity bounds.

Computer printout of some of this work is given here. Five batch programs were used and 1 is given here. In a batch program the instructions are read from a previously prepared file and executed one after the other. The instructions appear on the C lines in lower case, while MACSYMA's responses

appear on the D lines. Numerous explanatory comments are given and have the form:

/* This is a comment */.

Batch program A shows the derivation, solution, and verification of the 2x2 matrix Riccati equation associated with the 1-D dynamical translation white-noise force problem. The matrix Riccati equation appears in lines (C62) and (D62). The solutions are given by (D55), (D56), (D57), and (D58) where D(t) is defined by (D48). Derivation is given by the preceding steps, while verification is given in subsequent steps. The matrices $\underset{\approx}{A}$, $\underset{\approx}{M}$, and $\underset{\approx}{N}$ used in defining the Riccati equation are defined at the beginning of the printout. Note that the form of the Riccati equation assumes that the time is normalized in a suitable way (with no loss of generality). Note the symmetry of the matrix elements as required. A rough computer plot of the matrix elements is also given.

```
(C8) plot([r11,r12,r22],t,0,3,"Dimensionless time t - (tmax = 3)","V11*  V12#  V22$",["*","#","$"]);
```

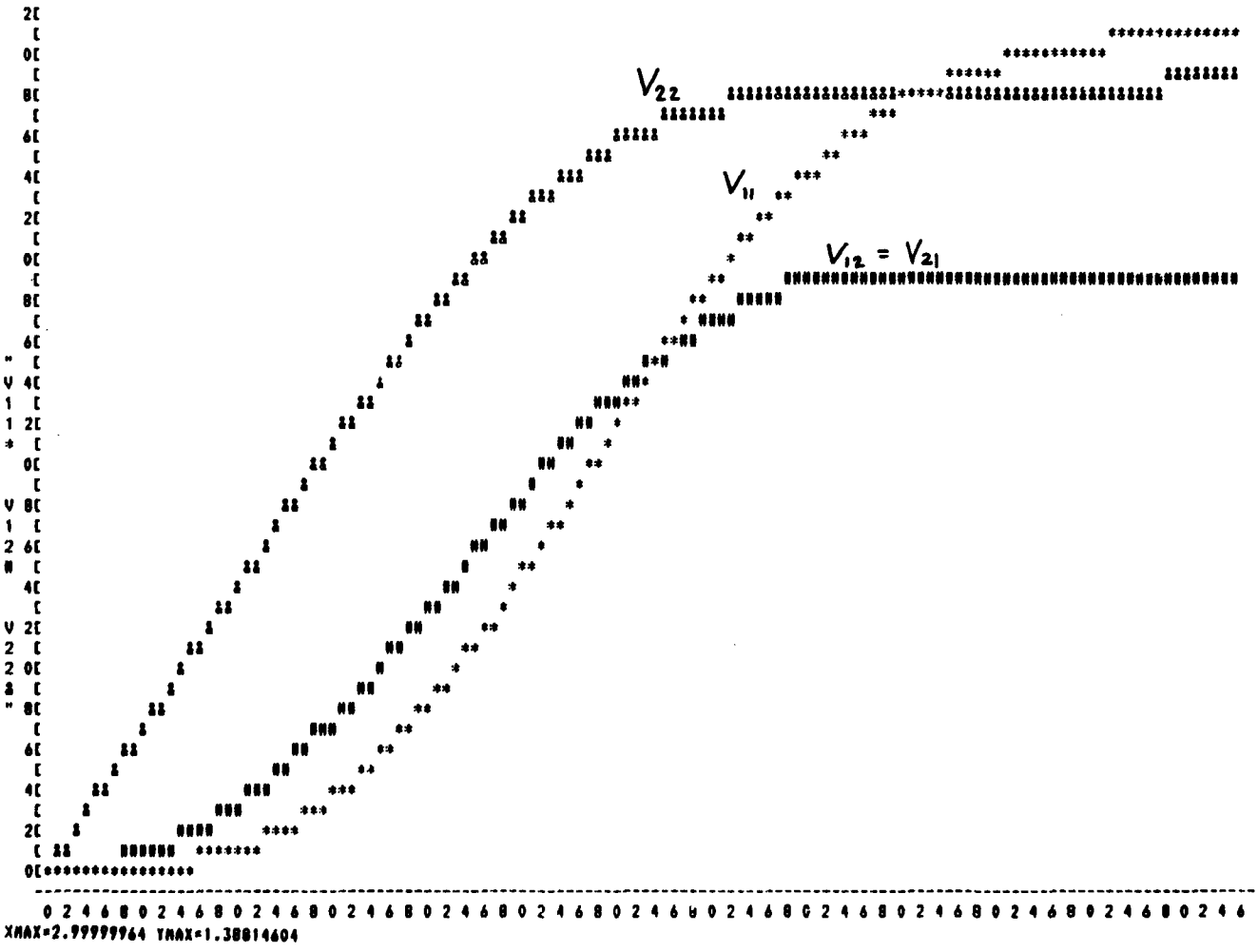
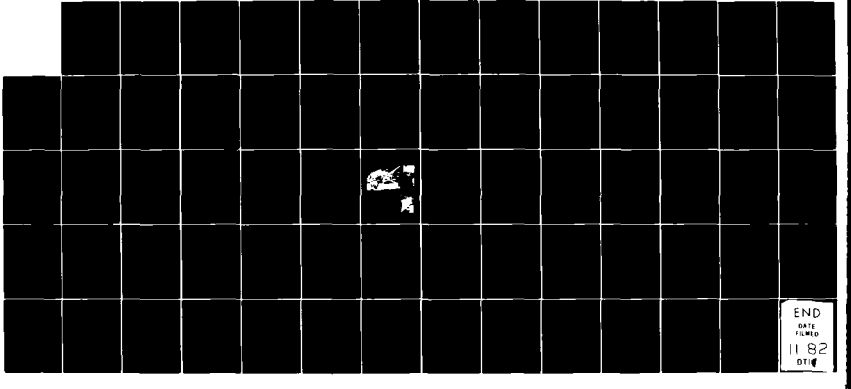


FIGURE A-1. COVARIANCE MATRIX

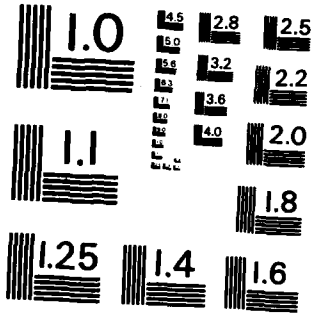
AD-A121 018 ANALYTIC PERFORMANCE MODELING OF IMAGING TRACKERS(U) BATTELLE COLUMBUS LABS OH R L FREY ET AL. 05 MAR 82 MAC-FR82-72-784 DAAG29-76-D-0100

2/2

UNCLASSIFIED F/G 20/6 NL



END
DATE FILMED
11 82
DTIC



BATCH PROGRAM A

```
batch(ric22)$

(C2) batch(ric22a,>,dsk,wilcox)$

(C3) /* Definitions of matrices for the 2*2 Riccati equation */

/* Define A */

a:matrix(2,2,1,1,2);
Time= 46 msec.
```

$$(B3)\qquad \begin{bmatrix} 0 & 1 \\ & \\ 0 & 0 \end{bmatrix}$$

```
(C4) /* Define N */

n:matrix(2,2,f,2,2);
Time= 5 msec.
```

$$(B4)\qquad \begin{bmatrix} 0 & 0 \\ & \\ 0 & F \end{bmatrix}$$

```
(C5) /* Define M */

m:matrix(2,2,1/f,1,1);
Time= 11 msec.
```

$$(B5)\qquad \begin{bmatrix} 1 & \\ - & 0 \\ F & \\ & \\ 0 & 0 \end{bmatrix}$$

```
(C6) /* Define B */

b:matrix([a,n],[m,-transpose(a)]);
Time= 31 msec.
```

$$(B6)\qquad \begin{bmatrix} \begin{bmatrix} 0 & 1 \\ & \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ & \\ 0 & F \end{bmatrix} \\ \begin{bmatrix} 1 & \\ - & 0 \\ F & \\ & \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ & \\ - 1 & 0 \end{bmatrix} \end{bmatrix}$$

```
(C7) bimatrix([0,1,0,0],[0,0,0,f],[1/f,0,0,0],[0,0,-1,0]);
Time= 13 msec.
```

$$(D7)\quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & F \\ 1 & & & \\ - & 0 & 0 & 0 \\ F & & & \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

```
(C8) /* Define X */

x:matrix([x11(t),x12(t)],[x21(t),x22(t)]);
Time= 9 msec.
```

$$(D8)\quad \begin{bmatrix} X11(T) & X12(T) \\ X21(T) & X22(T) \end{bmatrix}$$

```
(C9) /* Define Y */

y:matrix([y11(t),y12(t)],[y21(t),y22(t)]);
Time= 9 msec.
```

$$(D9)\quad \begin{bmatrix} Y11(T) & Y12(T) \\ Y21(T) & Y22(T) \end{bmatrix}$$

```
(C10) /* Define Z */

z:matrix([x11(t),x12(t)],[x21(t),x22(t)],[y11(t),y12(t)],[y21(t),y22(t)]);
Time= 16 msec.
```

$$(D10)\quad \begin{bmatrix} X11(T) & X12(T) \\ X21(T) & X22(T) \\ Y11(T) & Y12(T) \\ Y21(T) & Y22(T) \end{bmatrix}$$

```
Time= 1525 msec.

(C12) batch(ric22b,>,dsk,wilcox)$
```

```
(C13) /* Obtain differential eqs. for Z*/

zeq:diff(z,t) = b.z;
Time= 702 msec.
```

$$
\begin{bmatrix} \frac{d}{dT}(X11(T)) & \frac{d}{dT}(X12(T)) \\ \frac{d}{dT}(X21(T)) & \frac{d}{dT}(X22(T)) \\ \frac{d}{dT}(Y11(T)) & \frac{d}{dT}(Y12(T)) \\ \frac{d}{dT}(Y21(T)) & \frac{d}{dT}(Y22(T)) \end{bmatrix} = \begin{bmatrix} X21(T) & X22(T) \\ F\,Y21(T) & F\,Y22(T) \\ X11(T) & X12(T) \\ \frac{}{F} & \frac{}{F} \\ -Y11(T) & -Y12(T) \end{bmatrix} \tag{D13}
$$

```
(C14) eqs:[]$
Time= 4 msec.

(C15) for i:1 thru 4 do (for j:1 thru 2 do eqs:cons(part(zeq,1,i,j)=part(zeq,2,i,j),eqs))$
Time= 162 msec.

(C16) /* Specify  unknowns to be solved for */

un:[]$
Time= 4 msec.

(C17) for i:1 thru 4 do (for j:1 thru 2 do un:cons(z[i,j],un))$
Time= 132 msec.

(C18) /* Set initial conditions */

xinit:subst(0,t,x)=ematrix(2,2,0,1,1);
Time= 13 msec.
```

$$
\begin{bmatrix} X11(0) & X12(0) \\ X21(0) & X22(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{D18}
$$

```
(C19) yinit:subst(0,t,y)=ident(2);
Time= 13 msec.
```

$$
\begin{bmatrix} Y11(0) & Y12(0) \\ Y21(0) & Y22(0) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{D19}
$$

```
(C20) for i:1 thru 2 do (for j:1 thru 2 do atvalue(x[i,j],t=0,part(xinit,2,i,j)))$
```

```
(C21) for i:1 thru 2 do (for j:1 thru 2 do atvalue(y[i,j],t=0,part(yinit,2,i,j)))$
Time= 83 msec.

Time= 3079 msec.

(C23) batch(ric22c,),dsk,wilcox)$

(C24) /* Solve Differential Eqs. for X and Y */

ans:desolve(eqs,un)$
Time= 5115 msec.

(C25) ans:subst((lvar^2+%i)*(lvar^2-%i),lvar^4+1,ans)$
Time= 123 msec.

(C26) for i:1 thru 8 do ans1[i]:part(ans[i],1)=ilt(part(ans[i],2,1),lvar,t)$
Time= 3657 msec.

(C27) for i:1 thru 8 do ans1[i]:ratexpand(ratsubst(t/sqrt(2),sqrt(2)*t/2,realpart(ans1[i])))$
Time= 26771 msec.

(C28) for i:1 thru 8 do print(ans1[i])$
                  T             T
Y22(T) = COS(-------) COSH(-------)
               SQRT(2)       SQRT(2)
                  T             T                T             T
            COS(-------) SINH(-------)   COSH(-------) SIN(-------)
               SQRT(2)       SQRT(2)        SQRT(2)       SQRT(2)
Y21(T) = - ------------------------- - -------------------------
                     SQRT(2)                       SQRT(2)
                  T             T                T             T
           COSH(-------) SIN(-------)   COS(-------) SINH(-------)
               SQRT(2)       SQRT(2)        SQRT(2)       SQRT(2)
Y12(T) = ------------------------- - -------------------------
                     SQRT(2)                       SQRT(2)
                  T             T
Y11(T) = COS(-------) COSH(-------)
               SQRT(2)       SQRT(2)
                  T             T                T             T
           F COS(-------) SINH(-------)   F COSH(-------) SIN(-------)
               SQRT(2)       SQRT(2)          SQRT(2)       SQRT(2)
X22(T) = ------------------------- + -------------------------
                     SQRT(2)                       SQRT(2)
                   T             T
X21(T) = - F SIN(-------) SINH(-------)
                SQRT(2)       SQRT(2)
                   T             T
X12(T) = F SIN(-------) SINH(-------)
               SQRT(2)       SQRT(2)
```

```
                    T               T                 T               T
         F COS(--------) SINH(--------)   F COSH(--------) SIN(--------)
                 SQRT(2)         SQRT(2)           SQRT(2)         SQRT(2)
X11(T) = ------------------------------ - ------------------------------
                    SQRT(2)                            SQRT(2)
Time= 847 msec.

(C29) /* Obtain X and Y matrices */

x:subst([ans1[5],ans1[6],ans1[7],ans1[8]],x)$
Time= 50 msec.

(C30) y:subst([ans1[1],ans1[2],ans1[3],ans1[4]],y)$
Time= 51 msec.

(C31) print("X = ",x)$
```

$$X = \begin{bmatrix} \dfrac{F \cos\left(\frac{T}{\sqrt{2}}\right) \sinh\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} - \dfrac{F \cosh\left(\frac{T}{\sqrt{2}}\right) \sin\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} & F \sin\left(\frac{T}{\sqrt{2}}\right) \sinh\left(\frac{T}{\sqrt{2}}\right) \\ - F \sin\left(\frac{T}{\sqrt{2}}\right) \sinh\left(\frac{T}{\sqrt{2}}\right) & \dfrac{F \cos\left(\frac{T}{\sqrt{2}}\right) \sinh\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} + \dfrac{F \cosh\left(\frac{T}{\sqrt{2}}\right) \sin\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} \end{bmatrix}$$

```
Time= 415 msec.

(C32) print("Y = ",y)$
```

$$Y = \begin{bmatrix} \cos\left(\frac{T}{\sqrt{2}}\right) \cosh\left(\frac{T}{\sqrt{2}}\right) & \dfrac{\cosh\left(\frac{T}{\sqrt{2}}\right) \sin\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} - \dfrac{\cos\left(\frac{T}{\sqrt{2}}\right) \sinh\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} \\ - \dfrac{\cos\left(\frac{T}{\sqrt{2}}\right) \sinh\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} - \dfrac{\cosh\left(\frac{T}{\sqrt{2}}\right) \sin\left(\frac{T}{\sqrt{2}}\right)}{\sqrt{2}} & \cos\left(\frac{T}{\sqrt{2}}\right) \cosh\left(\frac{T}{\sqrt{2}}\right) \end{bmatrix}$$

```
Time= 395 msec.

Time= 41343 msec.

(C34) batch(ric22d,>,dsk,wilcox)$
```

```
(C35) /* Obtain inverse of Y */

detout:true$
Time= 3 msec.

(C36) doallmxops:false$
Time= 3 msec.

(C37) yinv:y^^-1$
Time= 273 msec.

(C38) /* Obtain Determinant Denominator */

det:denom(yinv)$
Time= 4 msec.

(C39) /* Define substitutions */

sub1:sinh(t/sqrt(2))=(exp(t/sqrt(2))-exp(-t/sqrt(2)))/2;
Time= 761 msec.
```

$$(D39) \qquad \sinh\left(\frac{T}{\sqrt{2}}\right) = \frac{XE^{\frac{T}{\sqrt{2}}} - XE^{-\frac{T}{\sqrt{2}}}}{2}$$

```
(C40) sub2:cosh(t/sqrt(2))=(exp(t/sqrt(2))+exp(-t/sqrt(2)))/2;
Time= 186 msec.
```

$$(D40) \qquad \cosh\left(\frac{T}{\sqrt{2}}\right) = \frac{XE^{\frac{T}{\sqrt{2}}} + XE^{-\frac{T}{\sqrt{2}}}}{2}$$

```
(C41) sub3:(sin(t/sqrt(2)))^2;
Time= 124 msec.
```

$$(D41) \qquad \sin^2\left(\frac{T}{\sqrt{2}}\right)$$

```
(C42) sub3rs:1-(cos(t/sqrt(2)))^2;
Time= 128 msec.
```

$$(D42) \qquad 1 - \cos^2\left(\frac{T}{\sqrt{2}}\right)$$

```
(C43) sub4:%e^(2*t/sqrt(2))=(cosh(t/sqrt(2))+sinh(t/sqrt(2)))^2;
Time= 221 msec.
                                                2 T
                                               -------
                                               SQRT(2)            T              T     2
(D43)                                       %E         = (SINH(-------) + COSH(-------))
                                                              SQRT(2)        SQRT(2)

(C44) sub5:%e^(-2*t/sqrt(2))=(cosh(t/sqrt(2))-sinh(t/sqrt(2)))^2;
Time= 227 msec.
                                                2 T
                                              - -------
                                               SQRT(2)            T              T     2
(D44)                                       %E         = (COSH(-------) - SINH(-------))
                                                              SQRT(2)        SQRT(2)

(C45) sub6:(sinh(t/sqrt(2)))^2;
Time= 90 msec.
                                                                  2    T
(D45)                                                          SINH (-------)
                                                                    SQRT(2)

(C46) sub6r:(cosh(t/sqrt(2)))^2-1;
Time= 93 msec.
                                                                  2    T
(D46)                                                          COSH (-------) - 1
                                                                    SQRT(2)

(C47) /* Make above substitutions in Determinant */

den:ratsubst(sub6r,sub6,ratexpand(subst(sub4,subst(sub5,ratexpand(ratsubst(sub3r,sub3),ratexpand(subst([sub1,sub2],det)))))))$
Time= 3585 msec.

(C48) dnq:d(t) = den;
Time= 5 msec.
                                                        2    T          2    T
(D48)                                          D(T) = COSH (-------) + COS (-------)
                                                             SQRT(2)         SQRT(2)

(C49) /* Obtain Numerator of Y inverse */

yinvnum:num(yinv)$
Time= 5 msec.

(C50) /* Obtain Numerator of V = X . (Y inverse) */

vnum:x.yinvnum$
```

```
(C51) /* Simplify Vnumerator using same substitutions used to simplify D(t) */

$
(C51) for i:1 thru 2 do (for j:1 thru 2 do vnum[i,j]:ratexpand(ratsubst(sub3r,sub3l,ratexpand(subst([sub1,sub2],vnum[i,j])))))$
Time= 16317 msec.

(C52) for i:1 thru 2 do (for j:1 thru 2 do vnum[i,j]:ratsubst(sub6r,sub6l,ratexpand(subst(sub4,subst(sub5,vnum[i,j])))))$
Time= 4455 msec.

(C53) /* The solution to the 2*2 Matrix Riccati Equation is now given by */

vmatrix([v11(t),v12(t)],[v21(t),v22(t)])$
Time= 10 msec.

(C54) print("V = ",v)$
     [ V11(T)  V12(T) ]
V =  [                ]
     [ V21(T)  V22(T) ]
Time= 51 msec.

(C55) /* where V11(t),V12(t),V21(t),V22(t) are given by */

eq11:v11(t)=vnum[1,1]/d(t);
Time= 15 msec.

                                                T          T          T          T
                                    2 F COSH(-------) SINH(-------) - 2 F COS(-------) SIN(-------)
                                               SQRT(2)    SQRT(2)    SQRT(2)    SQRT(2)
(D55)                     V11(T) = ------------------------------------------------------------
                                                      SQRT(2) D(T)

(C56) eq12:v12(t)=vnum[1,2]/d(t);
Time= 15 msec.

                                                  2    T          2    T
                                              F COSH (-------) - F COS (-------)
                                                     SQRT(2)         SQRT(2)
(D56)                                 V12(T) = ---------------------------------
                                                          D(T)

(C57) eq21:v21(t)=vnum[2,1]/d(t);
Time= 15 msec.

                                                  2    T          2    T
                                              F COSH (-------) - F COS (-------)
                                                     SQRT(2)         SQRT(2)
(D57)                                 V21(T) = ---------------------------------
                                                          D(T)
```

```
(C58) eq22:v22(t)=vnum[2,2]/d(t);
Time= 15 msec.
                                           T                T                   T              T
                          2 F COSH(-------) SINH(-------) + 2 F COS(-------) SIN(-------)
                                       SQRT(2)          SQRT(2)             SQR(2)         SQRT(2)
(D58)                 V22(T) = --------------------------------------------------------------------
                                                         SQRT(2) D(T)

(C59) /* where D(t) is defined above */

/* Note that V is a symmetric matrix as required */


Time= 37566 msec.

(C60) batch(ric22e,>,dsk,wilcox)$

(C61) /* Verify that V satisfies the 2*2 Riccati equation */

/* First substitute V12(t) for V21(t) in V matrix */

print("V = ",v:subst(v12(t),v21(t),v))$
     [ V11(T)  V12(T) ]
V =  [                ]
     [ V12(T)  V22(T) ]
Time= 58 msec.

(C62) riceq:diff(v,t)=a.v+v.transpose(a)+n-v.n.v;
Time= 141 msec.
                                                   [                  2                                  ]
                 [ d              d            ]   [               V11 (T)                V11(T) V12(T)  ]
                 [ -- (V11(T))    -- (V12(T))  ]   [  2 V12(T) - -------      V22(T) - -------------   ]
                 [ dT             dT           ]   [                 F                          F        ]
(D62)            [                             ] = [                                                     ]
                 [ d              d            ]   [                                             2       ]
                 [ -- (V12(T))    -- (V22(T))  ]   [             V11(T) V12(T)              V12 (T)      ]
                 [ dT             dT           ]   [ V22(T) - -------------         F - -------          ]
                                                   [                 F                        F          ]

(C63) ric:[]$
Time= 4 msec.

(C64) for i:1 thru 2 do (for j:i thru 2 do ric:cons(part(riceq,1,i,j)=part(riceq,2,i,j),ric))$
Time= 72 msec.
```

```
(C65) for i:1 thru 3 do print(ric[i])$
                       2
d                 V12  (T)
-- (V22(T)) = F - --------
dT                   F
d                        V11(T) V12(T)
-- (V12(T)) = V22(T) - -------------
dT                            F
                               2
d                        V11  (T)
-- (V11(T)) = 2 V12(T) - -------
dT                          F
Time= 200 msec.

(C66) /* Differentiate the Solution */

soln:[eq22,eq12,eq11]$
Time= 5 msec.

(C67) dsoln:diff(soln,t)$
Time= 1647 msec.

(C68) for i:1 thru 3 do print(dsoln[i])$
                     2     T                  2     T                  2     T                    2     T
              2 F SINH (-------)   2 F SIN (-------)   2 F COSH (-------)    2 F COS (-------)
                         SQRT(2)             SQRT(2)              SQRT(2)               SQRT(2)
              ------------------ - ----------------- + ------------------ + -----------------
d                  SQRT(2)              SQRT(2)             SQRT(2)              SQRT(2)
-- (V22(T)) = -------------------------------------------------------------------------------
dT                                         SQRT(2) D(T)

                                                 T              T                     T              T        d
                                    (2 F COSH(-------) SINH(-------) + 2 F COS(-------) SIN(-------)) (-- (D(T)))
                                              SQRT(2)        SQRT(2)               SQRT(2)        SQRT(2)    dT
                                  - -------------------------------------------------------------------------------
                                                                                 2
                                                                     SQRT(2) D (T)
                         T              T                   T              T
              2 F COSH(-------) SINH(-------)   2 F COS(-------) SIN(-------)
                       SQRT(2)        SQRT(2)           SQRT(2)        SQRT(2)                2     T                  2     T        d
              ------------------------------- + -------------------------------   (F COSH (-------) - F COS (-------)) (-- (D(T)))
d                       SQRT(2)                            SQRT(2)                            SQRT(2)             SQRT(2)    dT
-- (V12(T)) = ----------------------------------------------------------------- - ---------------------------------------------------
dT                                         D(T)                                                          2
                                                                                                        D (T)
```

```
                                2    T              2    T              2    T              2    T
                           2 F SINH (-------)   2 F SIN (-------)   2 F COSH (-------)   2 F COS (-------)
                                 SQRT(2)             SQRT(2)             SQRT(2)             SQRT(2)
                           ------------------ + ------------------ + ------------------ - ------------------
d                               SQRT(2)             SQRT(2)             SQRT(2)             SQRT(2)
-- (V11(T)) = ------------------------------------------------------------------------------------------
dT                                                SQRT(2) D(T)

                                                                  T              T              T              T      d
                                                       (2 F COSH(-------) SINH(-------) - 2 F COS(-------) SIN(-------)) (-- (D(T)))
                                                              SQRT(2)        SQRT(2)        SQRT(2)        SQRT(2)    dT
                                                     - ------------------------------------------------------------------------------
                                                                                      2
                                                                               SQRT(2) D (T)
```

Time= 1144 msec.

(C69) /* Verification proceeds by subtracting above set of equations from these */

/* and making appropriate substitutions and simplifications */

```
for i:1 thru 3 do ric[i]:subst(soln,dsolm[i]-ric[i])$
```

Time= 442 msec.

(C70) /* Obtain derivative of determinant */

```
dddt:diff(dnq,t);
```

Time= 184 msec.

```
                                                    T              T              T              T
                                           2 COSH(-------) SINH(-------)   2 COS(-------) SIN(-------)
                                                 SQRT(2)        SQRT(2)        SQRT(2)        SQRT(2)
                                d                ------------------------------ - ------------------------------
(D70)                           -- (D(T)) = SQRT(2)                            SQRT(2)
                                dT
```

```
(C71) for i:1 thru 3 do ric[i]:ratsubst(sub6r,sub6i,ratsubst(sub3r,sub3i,ratexpand(subst(dnq,ratsimp((d(t))^2*subst(dddt,ric[i]))))))
!
)$
```

Time= 7534 msec.

```
(C72) for i:1 thru 3 do print(ric[i])$
0 = 0
0 = 0
0 = 0
```

Time= 80 msec.

(C73) /* This completes the verification */

Time= 13033 msec.

Time= 96039 msec.

Appendix B - Representation of Rectangular Solids by Vertex Projections

The target model is assumed to be a rectangular solid. For the tracking problem, it is necessary to determine the silhouette of the target projected onto the image plane. Figure 1 defines the notation for the target model.

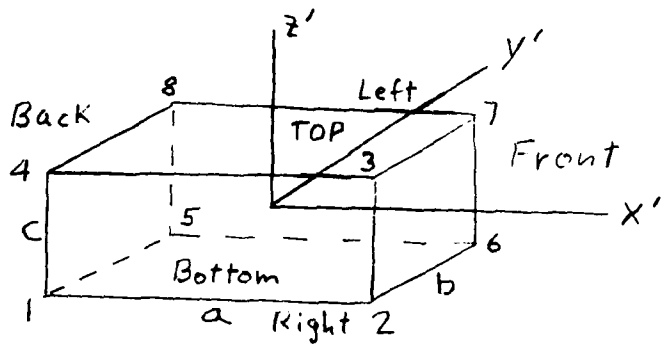


Figure 1

The x', y', and z' axes are fixed relative to the target with the original coinciding with the center of mass. The front of the tank points in the direction of the x' axis, while z' is vertical and y' points off to the left. It is assumed that at t = 0, the x', y', and z' axes coincide with earth-fixed x, y, and z axes, but that in general they will be translated and rotated with respect to each other. To simplify the analysis, it is assumed that z = 0 at all times. That is, the tank moves only on level ground. However, the position of the tracker with respect to the target is arbitrary. In general, the tank makes an angle θ with respect to its initial direction as shown in Figure 2.

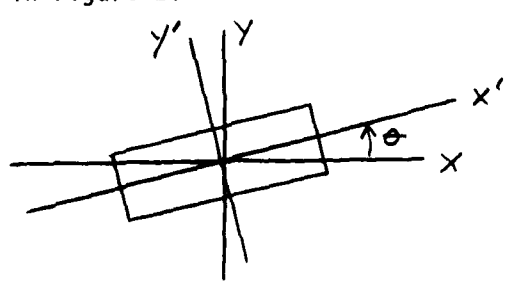


Figure 2. Orientation of Target as Seen From Above

Thus, the fixed and moving coordinate axes are related by

$$x = x_c + x' \cos\theta - y' \sin\theta$$

$$y = y_c + x' \sin\theta + y' \cos\theta$$

$$z = z'$$

where x_c and y_c are the x and y coordinates of the centroid.

While the position of the target is assumed to be given by

$$\vec{r} = x\hat{i} + y\hat{j} + z\hat{k},$$

where $\hat{i}$, $\hat{j}$, and $\hat{k}$ are an orthogonal set of unit vectors, the position of the tracker is assumed to be given by

$$\vec{R} = X\hat{i} + Y\hat{j} + Z\hat{k} .$$

The unit vector n defining the direction of the target as seen by the tracker is given by

$$\hat{n} = \frac{\vec{r} - \vec{R}}{|\vec{r} - \vec{R}|} .$$

We specialize further to the situation where the tracker and target are far from each other so that $\hat{n}$ is constant for all practical purposes as the target goes through its motion. In effect then, only the angular orientation affects the projected silhouette on the image plane.

In addition to $\hat{n}$, two orthogonal unit vectors $\hat{p}$ and $\hat{q}$ are defined by

$$\hat{p} \times \hat{n} = \hat{q} .$$

$\hat{p}$ and $\hat{q}$ define the direction of the "horizontal" and "vertical" axes for the tracker image plane.

The projection for each of the eight vertices onto the image plane is required. For any arbitrary vector $\vec{r}$, this projection $P\vec{r}$ is given by

$$P\vec{r} = (\hat{p}\cdot\vec{r})\hat{p} + (\hat{q}\cdot\vec{r})\hat{q}$$

$$\equiv P\hat{p} + q\hat{q} .$$

The unit vectors $\hat{n}$, $\hat{P}$, and $\hat{q}$ are related to the unit vectors $\hat{i}$, $\hat{j}$, and $\hat{k}$ by

$$\hat{n} = \hat{j} \cos \psi - \hat{k} \sin \psi$$

$$\hat{P} = \hat{i}$$

$$\hat{q} = \hat{P} \times \hat{n} = \hat{k} \cos \psi + \hat{j} \sin \psi$$

or

$$\hat{i} = \hat{P}$$

$$\hat{j} = \hat{n} \cos \psi + \hat{q} \sin \psi$$

$$\hat{k} = \hat{q} \cos \psi - \hat{n} \sin \psi$$

where ψ is the depression angle of the target as seen from the tracker.

$$\psi = \tan^{-1} \frac{Z}{-Y}$$

Thus, the "horizontal" and "vertical" components in the image plane, p and q, are given by

$$p = \hat{p} \cdot \vec{r} = \hat{i} \cdot \vec{r} = x$$

$$q = \hat{q} \cdot \vec{r} = y \sin \psi + z \cos \psi$$

Coordinates of x', y' and z' are as follows:

Point #	x'	y'	z'
1	-a/2	-b/2	-c/2
2	a/2	-b/2	-c/2
3	a/2	-b/2	c/2
4	-a/2	-b/2	c/2
5	-a/2	b/2	-c/2
6	a/2	b/2	-c/2
7	a/2	b/2	c/2
8	-a/2	b/2	c/2

From these coordinates, the projections onto the image plane are as follows:

$$p_1 = x_c - \tfrac{1}{2} a \cos\theta + \tfrac{1}{2} b \sin\theta$$

$$p_2 = x_c + \tfrac{1}{2} a \cos\theta + \tfrac{1}{2} b \sin\theta$$

$$p_3 = p_2$$

$$p_4 = p_1$$

$$p_5 = x_c - \tfrac{1}{2} a \cos\theta - \tfrac{1}{2} b \sin\theta$$

$$p_6 = x_c + \tfrac{1}{2} a \cos\theta - \tfrac{1}{2} b \sin\theta$$

$$p_7 = p_6$$

$$p_8 = p_5$$

$$q_1 = \sin\psi\, y_c - \tfrac{1}{2} a \sin\theta - \tfrac{1}{2} b \cos\theta - \tfrac{1}{2} c \cos\psi$$

$$q_2 = \sin\psi\, y_c + \tfrac{1}{2} a \sin\theta - \tfrac{1}{2} b \cos\theta - \tfrac{1}{2} c \cos\psi$$

$$q_3 = \sin\psi\, y_c + \tfrac{1}{2} a \sin\theta - \tfrac{1}{2} b \cos\theta + \tfrac{1}{2} c \cos\psi$$

$$q_4 = \sin\psi\, y_c - \tfrac{1}{2} a \sin\theta - \tfrac{1}{2} b \cos\theta + \tfrac{1}{2} c \cos\psi$$

$$q_5 = \sin\psi\, y_c - \tfrac{1}{2} a \sin\theta + \tfrac{1}{2} b \cos\theta - \tfrac{1}{2} c \cos\psi$$

$$q_6 = \sin\psi\, y_c + \tfrac{1}{2} a \sin\theta + \tfrac{1}{2} b \cos\theta - \tfrac{1}{2} c \cos\psi$$

$$q_7 = \sin\psi\, y_c + \tfrac{1}{2} a \sin\theta + \tfrac{1}{2} b \cos\theta + \tfrac{1}{2} c \cos\psi$$

$$q_8 = \sin\psi\, y_c - \tfrac{1}{2} a \sin\theta + \tfrac{1}{2} b \cos\theta + \tfrac{1}{2} c \cos\psi$$

Now it may be shown that the quantity

$$(q - q_i)(p_j - p_i) - (p - p_i)(q_j - q_i)$$

is zero on the straight line between the two points i and j, positive in the region to the left of the line in going from i to j, and negative in the region to the right of the line in going from i to j. (It is also easily shown that this quantity is invariant to coordinate translations and rotations.) Hence, the target silhouette can be defined in terms of the

following function

$$U_{ij} \equiv U_{ij}(p,q) = U((q - q_i)(p_j - p_i) - (p - p_i)(q_j - q_i))$$

where $U(x)$ is the unit step function

$$U(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

A different projection $U^{(k)}(p,q)$ is obtained depending upon which corner number k is closest to the tracker. One finds

$$U^{(3)}(p,q) = U_{12}\, U_{26}\, U_{67}\, U_{78}\, U_{84}\, U_{41}$$

$$U^{(4)}(p,q) = U_{51}\, U_{12}\, U_{23}\, U_{37}\, U_{78}\, U_{85}$$

$$U^{(7)}(p,q) = U_{26}\, U_{65}\, U_{58}\, U_{84}\, U_{43}\, U_{32}$$

$$U^{(8)}(p,q) = U_{65}\, U_{51}\, U_{14}\, U_{43}\, U_{37}\, U_{76}$$

where the dependence upon p and q has been suppressed. (These results have been derived with the aid of a three-dimensional solid rectangular model whose corners are numbered.)

These four quantities can be thrown into a single formula for the projection of the target on the image plane, valid for any orientation so long as

$$0 \leq \psi < 90^{\circ},$$

$$\begin{aligned} J(p,q) &= U(-\alpha)\, U(\beta)\, U^{(3)}(p,q) + U(-\alpha)\, U(-\beta)\, U^{(7)}(p,q) \\ &\quad + U(\alpha)\, U(-\beta)\, U^{(8)}(p,q) + U(\alpha)\, U(\beta)\, U^{(4)}(p,q) \\ &\equiv J(p,q;\, X_c,\, Y_c,\, \theta,\, \psi) \end{aligned}$$

where α and β are the direction cosines defined by

$$\alpha = \hat{n} \cdot \hat{i}'$$

$$\beta = \hat{n} \cdot \hat{j}'$$

Since

$$\hat{n} = \frac{\vec{r}_c - \vec{R}}{|\vec{r}_c - \vec{R}|}$$

where

$$\vec{r}_c - \vec{R} = x_c\,\hat{i} + (y_c + R\cos\psi)\,\hat{j} - R\sin\psi\,\hat{k}$$

and

$$\hat{i}' = \hat{i}\cos\theta + \hat{j}\sin\theta$$

$$\hat{j}' = \hat{j}\cos\theta - \hat{i}\sin\theta$$

$$\alpha' \equiv (\vec{r}_c - \vec{R})\cdot\hat{i}' = x_c\cos\theta + (y_c + R\cos\psi)\sin\theta$$

$$\beta' \equiv (\vec{r}_c - \vec{R})\cdot\hat{j}' = (y_c + R\cos\psi)\cos\theta - x_c\sin\theta$$

Since α' and β' have the same sign as α and β, respectively, one can also write $U(p,q;\, x_c,\, y_c,\, \theta,\, \psi)$

$$= U(-\alpha')\,U(\beta')\,U^{(3)}(p,q) + U(-\alpha')\,U(-\beta')\,U^{(3)}(p,q)$$

$$+ U(\alpha')\,U(-\beta')\,U^{(8)}(p,q) + U(\alpha')\,U(\beta')\,U^{(4)}(p,q)\ .$$

This result is valid for

$$0 \le \psi < 90^{o}.$$

For the case where $\psi = 90^{o}$, i.e., looking straight down on the targets,

$$U(p,q;\, x_c,\, y_c,\, \theta,\, 90^{o}) = U_{43}\,U_{37}\,U_{78}\,U_{84}\ .$$

APPENDIX C WHITE GAUSSIAN FORCING FUNCTION

This appendix fills in the details of the calculation for the 1-D translational-motion-only white-noise force model. One begins by converting the basic dynamical equation

$$m\ddot{x} = F_s + F_r(t) - K\dot{x}$$

into a first-order system in terms of state variables x_1 and x_2 defined relative to the average velocity F_s/k produced by the steady force F_s as follows:

$$x_1 = x - (F_s/K)t$$
$$x_2 = \dot{x}_1 - F_s/K$$

Thus

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = - Dx_2 + a_r(t)$$

where the damping constant D is defined by

$$D = k/m$$

and the random acceleration $a_r(t)$ is defined by

$$a_r(t) = F_r(t)/m$$

where $F_r(t)$ is the random white-noise force. Combining the state variables x_1 and x_2 into the state vector $\vec{X}$ defined by

$$\vec{X} = \begin{pmatrix} \vec{x}_1 \\ x_2 \end{pmatrix}$$

the stochastic dynamical equation for the system is given by

$$d\vec{x} = \underset{\approx}{A}\,\vec{x}\,dt + \underline{B}\,dw_T$$

where the dynamical matrix A is defined by

$$\underset{\approx}{A} = \begin{pmatrix} 0 & 1 \\ 0 & -D \end{pmatrix}$$

B is defined by $$\vec{B} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and dw_r is defined by

$$dw_r = a_r(t)\, dt$$

Since $a_r(t)$ is a white-noise process, $v_r(t)$ is a Brownian motion process satisfying

$$\langle dw_r dw_r \rangle = Q\, dt$$

This completes the statistical-dynamical specification of the state variables. To proceed further, it is necessary to specify the measurement process. The measurements are all assumed to be contained in an n-component column vector $\vec{g}(x)$

$$\vec{g}(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_n(x) \end{pmatrix}$$

where $g_i(x)$ is the temperature measured by the i^{th} pixel (in the absence of measurement noise) and it is assumed that the sensor has n pixels in all. Defining $\vec{z}$ as an n-component column vector whose components represent Brownian motions, i.e. $d\vec{z}$ represents a white-noise increment added in a time dt, the total observed increment to the measurement process is given by

$$d\vec{y} = \vec{g}(\vec{x})\, dt + d\vec{Z}$$

It is assumed that the noise process for different pixels are uncorrelated, so that the noise spectral-density matrix $\underset{\approx}{R}$ is given by

$$\underset{\approx}{R} = R\, \underset{\approx}{I}$$

where $\underset{\approx}{I}$ is the n x n idenity matrix, and R is the spectral density of any pixel, i.e.

$$\langle (dZ_i)^2 \rangle = R\, dt$$

for any pixel i.

The quantity $g_i(x)$ is defined by $g_i(x)=\beta_i(x)\,\Delta T$ where $\beta_i(x)$ is the fraction of the pixel area which is illuminated by the target's image. If the i^{th} pixel is completely inside the target's image when the target is at x then

$$\beta_i(x) = 1,$$

while if the pixel is completely outside

$$\beta_i(x) = 0$$

Now the general formalism involved linearizing the measurement vector about the estimate of the state vector, leads to a Riccati equation of the form

$$\dot{\underset{\approx}{V}} = \underset{\approx}{A}\,\underset{\approx}{V} + \underset{\approx}{V}\,\underset{\approx}{A}^t + \underset{\approx}{M} - \underset{\approx}{V}\,\underset{\approx}{N}\,\underset{\approx}{V}$$

where $\underset{\approx}{A}$ has been previously defined, $\underset{\approx}{M}$ is defined by

$$\underset{\approx}{M} = \vec{Q}\vec{B}\vec{B}^T$$

$$= \begin{pmatrix} 0 & 0 \\ 0 & q \end{pmatrix}$$

and $\underset{\approx}{N}$ is defined by

$$N_{12} = N_{21} = N_{22} = 0$$

$$N_{11} = \left[\frac{\partial \underset{\sim}{g}(\vec{x})}{\partial \vec{x}}\right]^T \underset{\approx}{R}^{-1} \frac{\partial \underset{\sim}{g}(\vec{x})}{\partial \vec{x}}$$

Since $\partial g_i(x)/\partial x = 0$ if the pixel i is not at a vertical boundary, and is equal to $\pm\,\Delta T/\varepsilon$ if it is, and since there are $2\,w/\delta$ such pixels it follows that

$$N_{11} = 2\,\frac{w}{\delta}\left(\frac{\Delta T}{\delta}\right)^2 \cdot \frac{1}{R}$$

consequently

$$\underset{\approx}{N} = \begin{pmatrix} q & 0 \\ 0 & 0 \end{pmatrix}$$

where

$$q = \frac{2\,W\,(\Delta T)^2}{\delta^3\,R}$$

$$= \frac{2\,W}{\delta^3}\,(SNR)^2\,B$$

It simplifies the solution some and gives some insight into how the relaxation time depends upon various parameters to convert to a dimensionless time variable equal to

$(Q\,q)^{1/4}$ times the regular time variable. When this is done, the Riccati equation has the same form except that the definitions of the $\underset{\approx}{A}$, $\underset{\approx}{M}$, and $\underset{\approx}{N}$ matrices

$$\underset{\approx}{A} = \begin{pmatrix} 0 & 1 \\ 0 & D' \end{pmatrix}$$

$$\underset{\approx}{M} = \begin{pmatrix} 0 & 0 \\ 0 & F_1 \end{pmatrix}$$

$$\underset{\approx}{N} = \begin{pmatrix} 1/F_1 & 0 \\ 0 & 0 \end{pmatrix}$$

where

$$D' = D/(Q\,q)^{1/4}$$

and

$$F_1 = (Q/q^3)^{1/4}$$

Under usual circumstances D' is small compared with 1, and therefore may be neglected.

The solution of the Riccati equation proceeds by solving the linear system

$$\dot{\underset{\approx}{X}} = \underset{\approx}{A}\,\underset{\approx}{X} + \underset{\approx}{M}\,\underset{\approx}{Y}$$

$$\dot{\underset{\approx}{Y}} = \underset{\approx}{N}\,\underset{\approx}{X} - \underset{\approx}{A}^{T}\,\underset{\approx}{Y}$$

subject to the initial conditions

$$\underset{\approx}{X}(o) = \underset{\approx}{0}$$

$$\underset{\approx}{Y}(o) = \underset{\approx}{I}$$

It may then be shown that a $\underset{\approx}{V}$ defined by

$$\underset{\approx}{V} = \underset{\approx}{X}\,\underset{\approx}{Y}^{-1}$$

satifies the given Riccati equation.

A MACSYMA program to solve this Riccati equation is given in Appendix A. (See batch program A, which contains explanatory comments for the derivation and verification. The solution is given by D55, D56, D57, and D58 together with the definition D48.)

Since as $t \to \infty$

$$V_{11} \to \sqrt{2\,F_1}$$

and

$$\sigma_{TRACK} = \sqrt{V_{11}} \to 2^{1/4}\,\sqrt{F_1}$$

$$= (4\,Q/q^3)^{1/8}$$

APPENDIX D - GENERALIZED POISSON FORCING FUNCTION

In this appendix it is shown how the method of linearization is used to approximate tracker RMS error when the 1-D rectangular target defined in section 5 which satisfies the 1-D dynamic equation

$$m\ddot{x} = F - K\dot{x} \tag{D-1}$$

where x is the target's location, k is a damping constant, and F is the random force. The random force, F, is assumed to consist of a d.c. term F_s and a zero-mean generalized Poisson term. That is,

$$F = F_s + F_r(t) \tag{D-2}$$

where F_s is the d.c. term and $F_r(t)$ is the generalized poisson term which undergoes random step changes of force with mean-square amplitude $\sigma^2_{F_r}$ and frequency λ_r

Written as a first order system, $x_j = x$, $s_2 = \dot{x}$, $x_3 = \ddot{x}$

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= a_s - Dx_2 + x_3 \\ \dot{x}_3 &= \dot{a}_r \end{aligned} \tag{D-3}$$

where $a_s \equiv F_s/m$

$a_r \equiv F_r/m$

$D \equiv K/m$

The quantity $x_3 = a_r$ is a generalized Poisson acceleration which undergoes a Brownian motion-type amplitude such that its spectral amplitude, Q, defined by

$$\left\langle (a_r(t))^2 \right\rangle = Q\,t \tag{D-4}$$

satisfies

$$Q = \lambda_r \sigma^2_{a_r} = \lambda_r \sigma^2_{F_r}/m^2 \tag{D-5}$$

where $\sigma^2_{a_r} = \sigma^2_{F_r}/m^2$ (the mean-square amplitude of the generalized Poisson acceleration).

From equation (D-3), the target's stochastic differential equation is

$$d\vec{x} = \underset{\approx}{A}\,\vec{X}\,dt + \underset{\approx}{B}\,d\,a_r \qquad \text{(D-6)}$$

where

$$\underset{\approx}{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -D & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\underset{\approx}{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and $a_r(t)$ is the generalized Poisson acceleration with spectral amplitude Q given in equation D-5.

The measurement is

$$d\vec{y} = \vec{g}(\vec{x})\,dt + d\vec{v} \qquad \text{(D-8}$$

where

g(x) is the noise free observation and

$$E\{d\vec{v}\,d\vec{v}^T\} = \underset{\approx}{R}\,dt\,. \qquad \text{(D-9)}$$

Linearization of (D-8) results in the following Riccati equation,

$$\dot{\underset{\approx}{V}} = \underset{\approx}{A}\,\underset{\approx}{V} + \underset{\approx}{V}\,\underset{\approx}{A}^T + \underset{\approx}{\sigma} - \underset{\approx}{V}\,\underset{\approx}{M}^T\,\underset{\approx}{R}^{-1}\,\underset{\approx}{M}\,\underset{\approx}{V} \qquad \text{(D-10)}$$

where

$$\sigma = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & p \end{pmatrix} \tag{D-11}$$

$$p = \lambda_r \sigma_{F_r}^2 / m^2 \tag{D-12}$$

and

$$\underset{\approx}{M} = \frac{\partial \vec{g}}{\partial \vec{x}} \tag{D-13}$$

The fact that $x_3(t) = a_r(t)$ behaves statistically like a Brownian motion implies that the time derivative

$$\dot{x}_3 = \dot{a}_r \tag{D-14}$$

behaves as a white-noise amplitude.

The matrix $\underset{\approx}{R} = [R_{ij}]$ is defined by the ensemble average as follows:

$$\left\langle V_i(t)V_j(t)\right\rangle = \int_0^t R_{ij}\,dt \tag{D-15}$$

In our case the i and j denote the various pixels of the detector. No correlation between different pixels is assumed, and it is also assumed that

$$R_{ij} = R \tag{D-16}$$

for each pixel.

Thus, it may be shown that

$$\underset{\approx}{N} = \underset{\approx}{M}^T \underset{\approx}{R}^{-1} \underset{\approx}{M} \tag{D-17}$$

has only one non-vanishing element

$$\underset{\approx}{N} = \begin{pmatrix} q^{-1} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{D-18}$$

where

$$q = \left[\frac{2\,W\,(\Delta T)^2}{\delta^3\,R}\right]^{-1} \tag{D-19}$$

Here W denotes the width of a rectangular target, δ is the pixel size, and ΔT is the contrast at a target boundary. (Note that $2W/\delta$ is the number of effective boundary pixels, while $\Delta T/\delta$ is the intensity slope at the detector image boundary due to finite pixel size)

It may be shown that the quantity R is related to noise equivalent temperature NEΔT, and the frame rate B, as follows:

$$R = \frac{(NE\Delta T)^2}{B} \tag{D-20}$$

Defining the signal-to-noise ratio SNR by

$$SNR = \frac{\Delta T}{NE\Delta T} \tag{D-21}$$

it follows that the matrix element q may be written

$$q = \frac{2\,W}{\delta^3}\,(SNR)^2\,B \tag{D-22}$$

Steady State Solution of the One-Dimensional Poisson Force Problem

The steady state solution of the Riccati equation is obtained by setting the time derivative of the variance matrix equal to zero and solving the resulting algebraic equations. MACSYMA has done this task for us, with the result that the tracker spatial error variance, V_{11}, is the solution of the following quartic equation:

$$(V_{11}\,q)^4 + 4\,D\,(V_{11}\,q)^3 + 4\,D^2\,(V_{11}\,q)^2 - 8\sqrt{P\,q}\,(V_{11}\,q) - 8\,D\sqrt{p\,q} = 0 \qquad \text{(D-23)}$$

while the other elements of V are given in terms of V_{11}, as follows:

$$\begin{aligned} V_{12} &= V_{21} = q\,V_{11}^2/2 \\ V_{13} &= V_{31} = \sqrt{p/q} \\ V_{22} &= (qV^3 + qD\,V_{11}^2)/2 - \sqrt{P/q} \\ V_{23} &= V_{32} = \sqrt{p\,q}\;V_{11} \\ V_{33} &= (Pq\,(V_{11}\,q + 2D)\,V_{11})/2 \end{aligned} \qquad \text{(D-24)}$$

Consider the example:

$$\begin{aligned} &\text{SNR} = 0.1 \\ &B = 15\ \text{hz} \\ &W = 10\ \text{ft.} \\ &W/_{\delta} = 10 \end{aligned} \qquad \text{(D-25)}$$

One finds

$$q = 3\ \text{ft}^2/\text{sec} \qquad \text{(D-26)}$$

Using the values related to the tank's stochastic dynamics (derived in a following section),

$$\begin{aligned} P &= 0.1334\ \text{ft}^2/\text{sec}^5 \\ D &= 0.025/\text{sec} \end{aligned} \qquad \text{(D-27)}$$

The quartic equation (D-23) has only one real solution,

$$V_{11} = 0.564 \text{ ft}^2$$

and
$$V_{12} = V_{21} = 0.477 \text{ ft}^2/\text{sec}$$
$$V_{13} = V_{31} = 0.211 \text{ ft}^2/\text{sec}^2$$
$$V_{22} = 0.609 \text{ ft}^2/\text{sec}^2 \qquad \text{(D-28)}$$
$$V_{23} = V_{32} = 0.357 \text{ ft}^2/\text{sec}^3$$
$$V_{33} = 0.311 \text{ ft}^2/\text{sec}^4$$

The RMS tracking error is given by

$$\sqrt{V_{11}} = 0.75 \text{ ft} \qquad \text{(D-29)}$$

i.e., the RMS tracking error is 75% of the target's width for these parameters.

Effect of Neglecting D

It is of interest to determine how much the results are changed if the damping factor D is set equal to zero. One finds

$$V_{11} = 2(P/q^5)^{1/6}$$
$$V_{12} = V_{21} = 2(P/q^2)^{1/3}$$
$$V_{13} = V_{31} = \sqrt{P/q} \qquad \text{(D-30)}$$
$$V_{22} = 3\sqrt{P/q}$$
$$V_{23} = V_{32} = 2(P^2/q)^{1/3}$$
$$V_{33} = 2(P^5/q)^{1/6}$$

Using the same values for p and q as above, one finds

$$
\begin{aligned}
V_{11} &= 0.572 \text{ ft}^2 \\
V_{12} &= V_{21} = 0.491 \text{ ft}^2/\text{sec} \\
V_{13} &= V_{31} = 0.211 \text{ ft}^2/\text{sec}^2 \\
V_{22} &= 0.633 \text{ ft}^2/\text{sec}^2 \\
V_{23} &= V_{32} = 0.362 \text{ ft}^2/\text{sec}^3 \\
V_{33} &= 0.311 \text{ ft}^2/\text{sec}^4
\end{aligned}
\qquad \text{(D-31)}
$$

Evidently the results are only slightly changed. In particular, the change in the RMS tracking error estimate is less than one percent.

The fact that V_{11} is proportional to the sixth-root of P, so that RMS tracking error depends upon the twelfth root of p, is fortunate since we probably can't estimate P too accurately. (Taking high roots of numbers brings them closer to 1.)

CHOICE OF THE DYNAMICAL CONSTANTS D AND P

The constant D is obtained from our model using data obtained from a tank-performance calculator. (See attachment). The calculator gives data on the time to reach 30 mph as a function of the tank's horsepower per ton ratio for two top speeds--38 mph and 63 mph.

The differential equation of motion may also be written

$$\frac{dv}{dt} = D(V_{max} - V)$$

where v is the velocity of the tank

and $V_{max} = \frac{F_{max}}{mD}$ is the tank's maximum velocity in terms of the maximum applied force F_{max}

Integrating the differential equation, assuming the tank starts from rest, and solving for D leads to the equation

$$D = \frac{1}{t} \ln\left(\frac{V_{max}}{V_{max-V}}\right)$$

The following table indicates results obtained.

	V_{max} = 38 mph		V_{max} = 63 mph	
HP/TON	t-sec	D $\frac{1}{sec}$	t-sec	D $\frac{1}{sec}$
30	12.9	0.121	12.4	0.052
40	9.5	0.164	9.0	0.072
50	7.7	0.202	7.2	0.090
60	7.7	0.236	6.0	0.108
70	5.8	0.268	5.3	0.122
80	5.2	0.300	4.7	0.138

The time t to obtain 30 miles per hour were obtained by reading the tank calculator, while the values for D were obtained from the above formula by setting V=30 and V_{max} equal to either 38 or 63.

To obtain P, we assume that σ_{F_r} is some reasonable fraction β(=1/2 say) of the maximum force F_{max}

$$\sigma_{F_r} = \beta F_{max} = F m D V_{max}$$

therefore

$$P = \lambda_r (\beta D V_{max})^2$$

Applying this to the Soviet T-62 medium tank (see enclosure), which has a 580 h.p. engine and weight of 41 tons, we thus get a HP/TON ratio of 14.15.

Extrapolating the tabular data down to this ratio, and assuming a top speed

$$V_{max} = 63 \text{ mph} = 92.4 \text{ ft/sec}$$

leads to

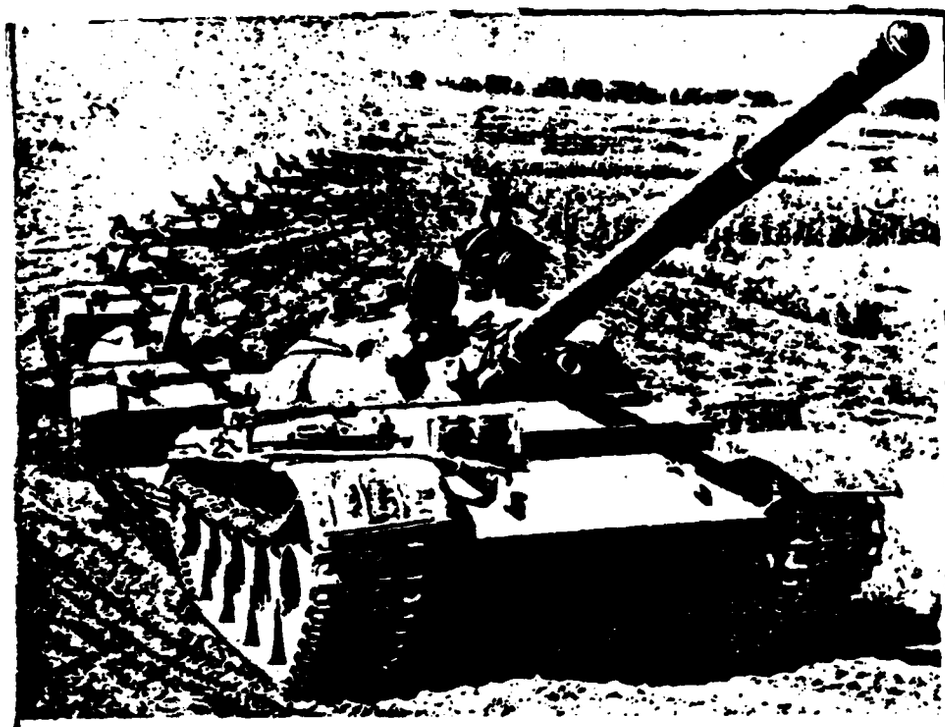
$$D = 0.025/\text{sec}$$

assuming the tank undergoes a step acceleration (or deceleration) on the average once very ten seconds, so that

$$\lambda_r = 0.1/\text{sec}$$

one finds with β chosen equal to 1/2 and V_{max} as above, that

$$P = 0.1334 \text{ ft}^2/\text{sec}^5$$



T-62 tanks snorkelling in Poland, no wider snorkel on the tank on the far b

A column of T-62s

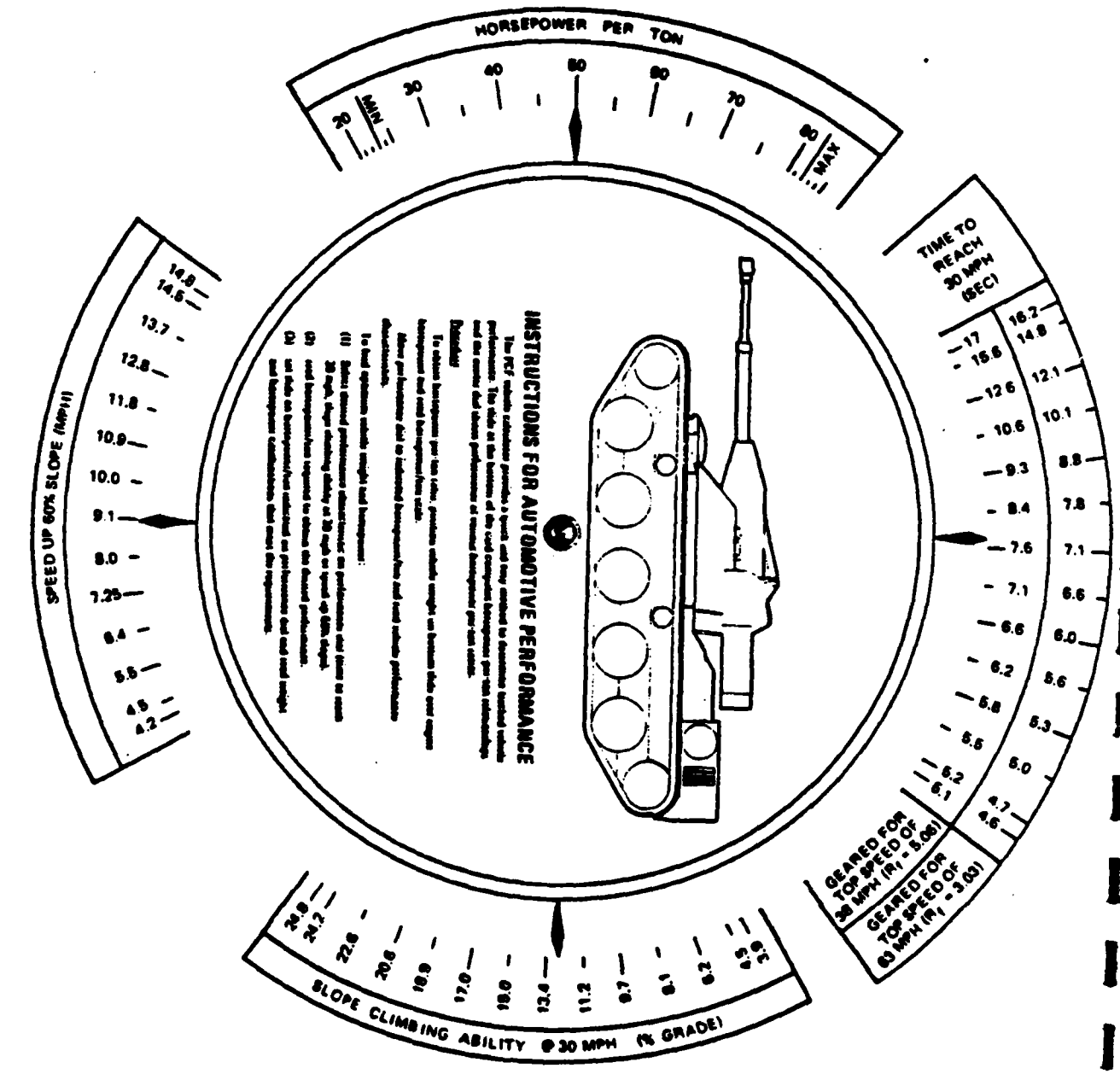
T-62 Medium Tank

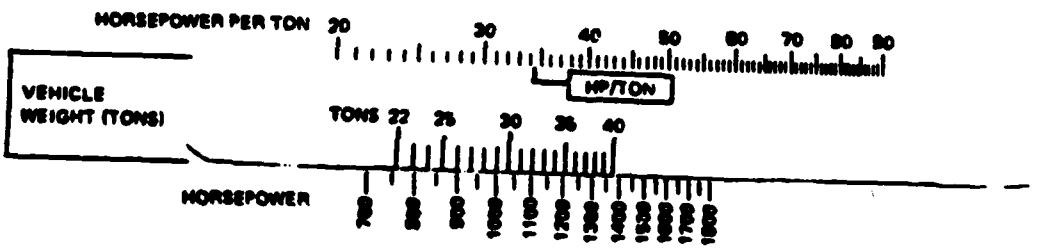
Soviet Union

Armament: One 115mm gun, elevation plus 15°, depression minus 3°, 44 rounds of 115mm ammunition are carried.
One 7.62mm machine gun co-axial with main armament.
Length: 31ft 0in (gun forward)
22ft 6in (vehicle)
Height: 7ft 10in
Weight: 82,000 lbs.
Range: 280 miles
Speed: 30 m.p.h. (road)
Fording: 4ft 6in, 13ft with snorkel
Engine: V-12 diesel developing 580 h.p.
Crew: 4
Width: 11ft 0in
G/Clearance: 16½in
G/Pressure: 11.40 psi
Gradient: 60%
V/Obstacle: 2ft 7in
Trench: 8ft 2in
Armour: 20-140mm
Development/Variants
The T-62 was shown to the public for the first time in May 1965, and it is believed to have gone into production in 1961/62. It is developed from the T-54 and T-55 series. The modifications include a longer and wider hull, new turret mounted slightly more to the rear, more track on the ground, no anti-aircraft or bow machine gun and revised turret hatches. The T-62 is armed with a new 115mm gun that fires fin-stabilised ammunition and may possibly be fitted with a laser rangefinder, the gun is longer that that fitted to the earlier T-54 and T-55 and has a bore evacuator.
The T-62 can be fitted with auxiliary fuel tanks at the rear of the vehicle, the T-62 can also lay its own smoke screen from its exhaust pipes on either side of the hull. Infra-red equipment is similar to that fitted to the T-55. The T-62 is easily recognisable from the T-54 and T-55 by its longer gun and the wider space between the roadwheels.
Employment
The T-62 is supplementing the T-54 and T-55 in the Soviet Army, it has also been supplied to Egypt and Czechoslovakia, it is reported that some have also been supplied to Poland and East Germany.

HIMAG

AUTOMOTIVE PERFORMANCE





APPENDIX E - DERIVATION OF BOBROVSKY-ZAKAI BOUND FOR A CLUTTERED SCENE

In this appendix, the Bobrovsky-Zakai (B-Z) bound is derived for the rectangular target model introduced in Section 5.1. The dynamic model, for both target and clutter, is

$$dX_T = \sqrt{q}\, dW \text{ (target motion)} \tag{E-1}$$

$$dX_C = 0 \text{ (clutter is at fixed location)} \tag{E-2}$$

where X_T and X_C are the target and clutter location, respectively; q is the targets velocity spectrum amplitude, and W is a normalized Wiener process.

The measurement equation is

$$d\vec{y} = \vec{h}(X_T, X_C)\, dt + \sqrt{r}\, d\vec{Z} \tag{E-3}$$

where $\vec{y}$ is a $N_R \cdot N_C \times 1$ vector (N_R is the number of rows and N_C is the number of columns) of pixel measurements. The i^{th} pixel measurement is given by

$$h_i(X_T, X_C) = T_T\, h_{T_i}(X_T)(1 - h_{C_i}(X_C)) + T_C\, h_{C_i}(X_C) \tag{E-4}$$

where

T_T = target intensity,

T_C = clutter intensity

$h_{C_i}(X_C)$ = clutter function given by

$$\frac{1}{\Delta^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \text{pixel}_i\,(\varepsilon_X, \varepsilon_Y)\ \text{rect}\left(\frac{\varepsilon_X - X_C}{\ell_C}\right) \text{rect}\left(\frac{\varepsilon_Y - h_C/2}{h_C}\right) \cdot d\varepsilon_X d\varepsilon_Y \quad \text{(E-5)}$$

where

Δ = pixel size

ℓ_C = length of clutter object

h_C = height of clutter object

$$\text{pixel}_i\,(\varepsilon_X, \varepsilon_Y) = \begin{cases} 1 \text{ if } \varepsilon_X, \varepsilon_Y \text{ on pixel } i \\ 0 \text{ otherwise} \end{cases}$$

and

$h_{T_i}(X_T)$ = target function

$$= \frac{1}{\Delta^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \text{pixel}_i\,(\varepsilon_X, \varepsilon_Y)\ \text{rect}\left(\frac{\varepsilon_X - X_T}{\ell_T}\right) \cdot \text{rect}\left(\frac{\varepsilon_Y - h_T/2}{h_T}\right) d\varepsilon_X d\varepsilon_Y \quad \text{(E-6)}$$

where

h_T = height of target object,

ℓ_T = length of target object.

In order to apply the B-Z bound to this problem we first compute

$$A_t = E \frac{\partial \vec{f}}{\partial \vec{x}} \tag{E-7}$$

where

$\vec{X} = \begin{bmatrix} X_T \\ X_C \end{bmatrix}$ and $\vec{f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ specifies the dynamic model. Then we solve for B in the equation

$$\begin{aligned} B^T(DD^T)^{-1}B &= E\left\{\left(\frac{\partial \vec{f}}{\partial \vec{x}} - \vec{A}\right)^T (CC^T)^{-1} \cdot \left(\frac{\partial \vec{f}}{\partial \vec{x}} - A\right)\right\} \\ &+ E\left\{\frac{\partial \vec{h}^T}{\partial \vec{x}} (DD^T)^{-1} \frac{\partial \vec{h}}{\partial \vec{x}}\right\} \end{aligned} \tag{E-8}$$

$C = \sqrt{q}$ specifies the random process which drives the dynamic model; $D = \sqrt{r}$ where $r = NE\Delta T/\Delta f$, Δf = the frame rate, and D specifies the noise level of the measurement process.

For the dynamic model of interest, equation (E-8) reduces to

$$\frac{B^T B}{r} = \frac{1}{r} E\left\{\frac{\partial \vec{h}^T}{\partial \vec{x}} \frac{\partial \vec{h}}{\partial \vec{x}}\right\} \tag{E-9}$$

Thus, only

$$E\left\{\frac{\partial \vec{h}^T}{\partial \vec{x}} \frac{\partial \vec{h}}{\partial \vec{x}}\right\} \tag{E-10}$$

need be evaluated in order to apply the B-Z bound. This is done by first noting that

$$E\left\{\frac{\partial \vec{h}^T}{\partial \vec{x}} \frac{\partial \vec{h}}{\partial \vec{x}}\right\} = \begin{bmatrix} E\frac{\partial \vec{h}^T}{\partial x_T} \frac{\partial \vec{h}}{\partial x_T} & E\frac{\partial \vec{h}^T}{\partial x_T} \frac{\partial \vec{h}}{\partial x_c} \\ E\frac{\partial \vec{h}^T}{\partial x_c} \frac{\partial \vec{h}}{\partial x_T} & E\frac{\partial \vec{h}^T}{\partial x_c} \frac{\partial \vec{h}_c}{\partial x_c} \end{bmatrix} \tag{E-11}$$

where

$$E\left(\frac{\partial \vec{h}^T}{\partial x_T} \frac{\partial \vec{h}}{\partial x_T}\right) = \sum_{i=1}^{N_R \cdot N_e} E\left(\frac{\partial h_i}{\partial x_T}\right)^2 \tag{E-12}$$

$$E\left(\frac{\partial \vec{h}^T}{\partial x_T} \frac{\partial \vec{h}}{\partial x_c}\right) = \sum_{i=1}^{N_R \cdot N_C} E\left\{\frac{\partial h_i}{\partial x_T} \frac{\partial h_i}{\partial x_c}\right\} \tag{E-13}$$

and

$$E\left(\frac{\partial \vec{h}^T}{\partial x_c} \frac{\partial \vec{h}}{\partial x_c}\right) = \sum_{i=1}^{N_R \cdot N_C} E\left(\frac{\partial h_i}{\partial x_c}\right)^2 \tag{E-14}$$

Thus at each pixel i the following expectations must be computed.

$$E\left\{\frac{\partial h_i}{\partial x_T}^2\right\} \tag{E-15}$$

$$E\left\{\frac{\partial h_i}{\partial x_c}^2\right\} \tag{E-16}$$

$$E\left\{\frac{\partial h_i}{\partial x_T}\frac{\partial h_i}{\partial x_C}\right\} \tag{E-17}$$

In order to evaluate equations (E-15) through (E-17) we note that $\frac{\partial h_i}{\partial x_T}$ will be non-zero only for those pixels on the targets left or right edge, and similarly, $\frac{\partial h_i}{\partial x_C}$ will be non-zero only for those pixels on the clutter object left or right edge; that is,

$$\frac{\partial h_i}{\partial x_T} = T_T \frac{\partial h_{T_i}}{\partial x_T}\left(1-h_{C_i}\right) \tag{E-18}$$

where

$$\frac{\partial h_{T_i}}{\partial x_T} = \frac{1}{\Delta}\begin{cases} -1 & \text{if pixel i is on right edge of target} \\ 1 & \text{if pixel i is on left edge of target} \\ 0 & \text{otherwise} \end{cases} \tag{E-19}$$

and

$$\begin{aligned}\frac{\partial h_i}{\partial x_C} &= -T_T\, h_{T_i}\frac{\partial h_{C_i}}{x_C} + T_C\frac{\partial h_{C_i}}{\partial x_C} \\ &= \frac{\partial h_{C_i}}{\partial x_C}\left(T_C - T_T\, h_{T_i}\right)\end{aligned} \tag{E-20}$$

where

$$\frac{\partial h_{C_i}}{\partial x_C} = \frac{1}{\Delta}\begin{cases} -1 & \text{if pixel i is on right edge of clutter object} \\ 1 & \text{if pixel i is on left edge of clutter object} \\ 0 & \text{otherwise} \end{cases} \tag{E-21}$$

For equation (E-15)

$$E\left\{\frac{\partial h_i}{\partial x_T}^2\right\} = T_T^2 \, E\left\{\frac{\partial h_{T_i}}{\partial x_T}^2 \left(1-h_{C_i}\right)^2\right\} \qquad \text{(E-22)}$$

Since the target and clutter locations are independent, equation (E-22) reduces to

$$E\left\{\frac{\partial h_i}{\partial x_T}^2\right\} = T_T^2 \, E\left\{\frac{\partial h_{T_i}}{\partial x_T}^2\right\} \cdot E\left\{1 - h_{C_i}\right\}^2 \qquad \text{(E-23)}$$

The two terms on the right hand side of equation (E-23) can be evaluated as follows: first we observe that

$$E\left(1 - h_{C_i}(X_C)\right)^2 = E\left(1 - 2h_{C_i}(X_C) + h_{C_i}{}^2(X_C)\right)$$

Since

$$E\, h_{C_i}(X_C) = 0 \cdot P \text{ (pixel i not on clutter)}$$

$$+ \; 1 \cdot P \text{ (pixel i is interior to the clutter object)}$$

$$+ \; 1/2 \cdot P \text{ (pixel i is on left or right edge of clutter object)}$$

$$= 1 \cdot \frac{\left(N_C - N_{P_C}\right)}{N_R \, N_{C_L}}$$

$$+ \; 1/2 \, \frac{N_{P_C}}{N_R \, N_{C_L}}$$

$$= \left(N_C - N_{P_C}/2\right) / \left(N_R \, N_{C_L}\right) \qquad \text{(E-24)}$$

where

N_C = number of pixels on clutter object,

N_{P_C} = number of pixels on left and right clutter edge

N_{CL} = max (h_T, h_C)

$$E\, h_{C_i}^2 = 1 \cdot \frac{\left(N_C - N_{P_C}\right)}{N_R\, N_{C_L}} + \frac{1}{4}\frac{N_{P_C}}{N_R\, N_{C_L}} \qquad \text{(E-25)}$$

$$= \frac{N_C}{N_R} - \frac{3}{4\, N_{C_L}} N_{P_C}$$

and

$$E\, h_{T_i}(X_T) = \frac{N_T}{N_R\, N_{C_L}} - \frac{N_{P_T}}{2\, N_R\, N_{C_L}} \qquad E\, h_{T_i}^2(X_T) = \frac{N_T}{N_R\, N_{C_L}} - \frac{3}{4\, N_R\, N_{C_L}} N_{P_T} \qquad \text{(E-26)}$$

where

N_{P_T} = number pixel on target left or right edge

we get

$$E\left(1 - h_{C_i}(X_C)\right)^2$$

$$= E\left(1 - 2\,h_i(X_C) + h_{C_i}{}^2(X_C)\right)$$

$$= 1 - 2\left(\frac{N_C}{N_R N_{C_L}} - \frac{N_{P_C}}{2 N_R N_{C_L}}\right) + \left(\frac{N_C}{N_R N_{C_L}} - \frac{3}{4 N_R N_{C_L}} N_{P_C}\right) \qquad \text{(E-27)}$$

$$= V_1\left(N_C, N_{P_C}\right)$$

For the first term of equation (E-26) we observe that from equation (E-19)

$$E\left\{\frac{\partial h_{T_i}{}^2}{\partial x_T}\right\} = \frac{1}{\Delta^2} \cdot p \text{ (pixel i is on left or right edge of target)} \qquad \text{(E-28)}$$

Thus

$$E\left\{\frac{\partial h_i{}^2}{\partial x_T}\right\} = \frac{T_T{}^2}{\Delta^2} \cdot V_1\left(N_C, N_{P_C}\right) \cdot P_T \qquad \text{(E-29)}$$

where P_T is the probability that a pixel is on the left or right target edge and $V_1\left(N_C, N_{P_C}\right)$ is as defined in equation (E-27).

Now for equation (E-16).

$$E\left\{\frac{\partial h_i{}^2}{\partial X_C}\right\} = E\left(\frac{\partial h_{C_i}}{\partial X_C}\right)^2 \cdot E\left(T_C - T_T\, h_{T_i}\right)^2 \qquad \text{(E-30)}$$

where we have again used the assumption that the target's and clutter's locations are independent. Using a procedure similar to that used to derive equation (E-29) we get

$$E\left\{\frac{\partial h_i^2}{\partial x_C}\right\} = 2 \cdot N_{P_C} \cdot \frac{1}{\Delta^2} \cdot V_2 \qquad \text{(E-31)}$$

where

$$V_2 = E\left(T_C - T_T\, h_{T_i}(X_T)\right)^2 = T_C^2 \left[1 - 2\frac{T_T}{T_C}\left(\frac{N_T}{N_R\, N_{CL}} - \frac{N_{P_T}}{2\, N_R\, N_{CL}}\right) + \left(\frac{N_T}{N_R\, N_{CL}} - \frac{3}{4\, N_R\, N_{CL}} N_{P_T}\right)\right]$$

and

N_{P_C} = number of clutter object pixels.

Rather than evaluating equation (E-17), we observed that there are an equal number of left and right edges on both the clutter and target object. Thus

$$\sum_{i=1}^{N_R \cdot N_C} E\left\{\frac{\partial h_i}{\partial x_T}\frac{\partial h_i}{\partial x_C}\right\} = \sum_{i=1}^{N_R \cdot N_C} E\left(\frac{\partial h_i}{\partial x_T}\right) E\left(\frac{\partial h_i}{\partial x_C}\right) = 0 \qquad \text{(E-32)}$$

We are now in a position to fill in the elements of the matrix of equation (E-11) by combining equations (E-12) through (E-14) with equations (E-29) through (E-32). This results in

$$E\,\dot{\bar{h}}^T\,\dot{\bar{h}} = \begin{bmatrix} T_T^2 \dfrac{2N_{P_T}}{\Delta^2} V_1\left(N_{P_C}, N_C\right) & 0 \\ 0 & \dfrac{2\,V_2\,N_{P_C}}{\Delta^2} \end{bmatrix} \tag{E-33}$$

where

$$V_1\left(N_{P_C}, N_C\right) = 1 - 2\left(\frac{N_C}{N_R\,N_{CL}} - \frac{N_{P_C}}{2N_R\,N_{CL}}\right) + \left(\frac{N_C}{N_R\,N_{CL}} - \frac{3}{4}\frac{N}{N_R\,N_{CL}}P_C\right) \tag{E-34}$$

and

$$V_2\left(N_T, N_{P_T}, T_T, T_C\right) = T_C^2\left[1 - 2\frac{T_T}{T_C}\left(\frac{N_T}{N_R\,N_{CL}} - \frac{N_{P_T}}{2N_R\,N_{CL}}\right) + \left(\frac{N_T}{N_R\,N_{CL}} - \frac{3}{4N_R\,N_{CL}}N_{P_T}\right)\right] \tag{E-35}$$

In order to proceed let

$$\alpha_T = (T_T^2)\frac{N_{P_T}}{\Delta^2}2V_1 \tag{E-36}$$

and

$$\alpha_C = \frac{2\,V_2\,N_{P_C}}{\Delta^2} \tag{E-37}$$

Then

$$E\,\tilde{h}^T\tilde{h} = \begin{bmatrix} \alpha_T & 0 \\ 0 & \alpha_C \end{bmatrix} \tag{E-38}$$

Thus a solution to equation (E-9) is

$$B = \begin{bmatrix} \sqrt{\alpha_T} & 0 \\ 0 & \sqrt{\alpha_C} \end{bmatrix} \tag{E-39}$$

Therefore the mean square error on

$$du = C\,dw \tag{E-40}$$

$$ds = Budt + \sqrt{r}\,dZ$$

given by

$V = E(\vec{u} - \hat{\vec{u}})^T (\vec{u} - \hat{\vec{u}})$ is a lower bound to the mean square error of the non-linear problem specified by equations (E-1) and (E-2).

Thus we only need to solve the matrix Riccati equation

$$\frac{dV}{dt} = \begin{bmatrix} q & 0 \\ 0 & 0 \end{bmatrix} - V \begin{bmatrix} \alpha_T & 0 \\ 0 & \alpha_C \end{bmatrix} V \tag{E-41}$$

For the variable of interest, i.e., the target location, we get for the mean square error:

$$V_{11}(t) = \sqrt{rq}/\alpha_T \; \frac{(1 - \exp(\sqrt{\alpha_T q/r}\; t)}{(1 + \exp(-\sqrt{\alpha_T q/r}\; t)} \tag{E-42}$$

where it has been assumed that the target and clutter locations were known at t = 0.

APPENDIX F PROGRAM LISTING

```
C MAIN ROUTINE FOR
C XSTEPSIMULATING OPTIMAT FILTER
C F (REPRESENTING DYNAMICS), AND H (REPRESENTING OUTPUTS)
C ARE TWO SUBROUTINES ENTERED BY USER AND LINKED BY USER

      CHARACTER*40 OUTFIL
      CHARACTER*5 ANS
      REAL*8 XJA,XJB
      COMMON/DATA/ZT(10000)
      DIMENSION ILW(7),IUP(7)
      DIMENSION BUPPER(7),BALOWER(7)
      DIMENSION XLOW(7),XUP(7),XSTEP(7),NSTEP(7),XZERO(7)
      COMMON/SPECS/NFOVX,NFOVY,DETX,DETY
      COMMON/TARGET/XSIZE,YSIZE,DELTAT,XTAR
      DIMENSION ZMN(7),ZSIG(7),ALOWER(7),UPPER(7)
      DIMENSION UNLWR(7),UNUPR(7),XX(7),STEP(7)
      COMMON/VARS/U(7),XJA(50000),XJB(50000)
      COMMON/NOISE/QM1(7),RM1(10000)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      DIMENSION BOFF(7),BSCL(7)
      DATA BOFF,BSCL/7*0.,7*1./
      DATA NSTEP/7*1/

      WRITE(6,1918)
 1918 FORMAT(1X,' INPUT SEED')
      READ(5,*)ISEED
      CALL SET_SEED(ISEED)

C SIMULATION INPUTS

      WRITE(6,135)
 135  FORMAT(1X,' OUTPUT FILE NAME?')
      READ(5,531)OUTFIL
 531  FORMAT(A40)
      OPEN(UNIT=20,NAME=OUTFIL,STATUS='NEW')
      WRITE(6,8811)
 8811 FORMAT(1X,' NUMBER OF TRIALS')
      READ(5,*)NTRIAL
      WRITE(6,2212)
 2212 FORMAT(1X,' TARGET SIZE X,Y')
      READ(5,*)XSIZE,YSIZE
      WRITE(6,2213)
 2213 FORMAT(1X,' FOV #X,#Y')
      READ(5,*)NFOVX,NFOVY
      NOBS=NFOVX*NFOVY
      WRITE(6,2214)
 2214 FORMAT(1X,' PIXEL SIZE X,Y')
      READ(5,*)DETX,DETY
      WRITE(6,2215)
 2215 FORMAT(1X,' DELTA-T')
      READ(5,*)DELTAT
      WRITE(6,1000)
 1000 FORMAT(1X,' ENTER NUMBER OF STATES')
      READ(5,*)NDIM
      WRITE(6,1010)
 1010 FORMAT(1X, 'ENTER Q INVERSE DIAGONAL ELTS ')
      READ(5,*)(QM1(I),I=1,NDIM)
      WRITE(6,1020)
 1020 FORMAT(1X,' ENTER R INVERSE ')
      READ(5,*)XRM1

      DO 2929 I=1,NOBS
      RM1(I)=XRM1
```

```
 2929   CONTINUE

        WRITE(6,1030)
 1030   FORMAT(1X,' ENTER NUMBER OF POINTS, RANGE<LOWER,UPPER>
     .  FOR EACH DIMENSION ')
        READ(5,*)(KLEN(I),XLOW(I),XUP(I),I=1,NDIM)
        WRITE(6,1040)
 1040   FORMAT(1X,' ENTER INITIAL UNIFORM DENSITY LIMITS
     .  <LOWER,UPPER> AND STEP SIZES ')
        READ(5,*)(UNLWR(I),UNUPR(I),XSTEP(I),I=1,NDIM)
        WRITE(6,1050)
 1050   FORMAT(1X,' ENTER NUMBER OF ITERATIONS ')
        READ(5,*)ITERS
        WRITE(6,1060)
 1060   FORMAT(1X,' WISH TO HAVE STATE PRINTOUT? ')
        ISWITCH=0
        READ(5,1070)IANS
 1070   FORMAT(A1)
        IF(IANS.EQ.'Y')ISWTCH=1
        WRITE(6,1080)
 1080   FORMAT(1X,' ENTER INITIAL STATE VAR VAL''S ')
        READ(5,*)(XZERO(I),I=1,NDIM)
        WRITE(6,123)
 123    FORMAT(1X,' HOW ABOUT A GAUSSIAN PRIOR?')
        READ(5,111)ANS
 111    FORMAT(A5)

C INITIALIZE NUMBER OF STEPS

        CALL NUMSTEP(XLOW,XUP,XSTEP,NDIM,NSTEP)

C START SIMULATION TRIALS

        DO 8888 ITRIAL=1,NTRIAL
        IF(ITRIAL.EQ.NTRIAL)ISWTCH=1

C INITIALIZE #

        CALL SET1(ALOWER,XLOW,1,NDIM)
        CALL SET1(UPPER,XUP,1,NDIM)
        CALL SET1(STEP,XSTEP,1,NDIM)
        CALL SET1(XX,XZERO,1,NDIM)

C INTIALIZE ARRAY

        CALL GETSCL(ALOWER,UPPER,SCALE,OFF)
        VAL=1.

C INITIALIZE UPPER AND LOWER BOUNDARIES

                DO 50 INDX1=1,7
                IF(INDX1.GT.NDIM)GOTO 40
                VAL=(UNUPR(INDX1)-UNLWR(INDX1))*VAL
                ILW(INDX1)=IFIX(SCALE(INDX1)*(UNLWR(INDX1)-OFF(INDX1)))+1
                IUP(INDX1)=IFIX(SCALE(INDX1)*(UNUPR(INDX1)-OFF(INDX1)))+1
                GOTO 50
 40             ILW(INDX1)=1
                SCALE(INDX1)=1.
                IUP(INDX1)=1
                KLEN(INDX1)=1
 50             CONTINUE

C SET UP INITIAL PROBABILITIES
```

```
        VAL=1./VAL
        CALL SETUP(XJA,VAL,ILW,IUP,ALOWER,UPPER,ANS)

C START ITERATIONS

                DO 1ØØ INDX2=1,ITERS

C UPDATE PROBABILITIES

                CALL FILTER(XX,QM1,RM1)
                CALL UPDATE(ALOWER,UPPER,STEP,ZMN,ZSIG)

C SWAP ARRAYS

                CALL SWITCH(XJA,XJB)
                WRITE(6,11ØØ)(IQ,ZMN(IQ),ZSIG(IQ),XX(IQ),IQ=1,NDIM)
                WRITE(2Ø,11Ø3)NDIM
 11Ø3           FORMAT(1X,' DIMENSION OF STATE SPACE=',I3)
                WRITE(2Ø,11ØØ)(IQ,ZMN(IQ),ZSIG(IQ),XX(IQ),IQ=1,NDIM)
 11ØØ           FORMAT(1X,' VARIABLE ',I3,' HAS MEAN ',F12.5,
                ' AND SIGMA ',F12.5,' IS ACTALLY ',F12.5)
                IF(ISWTCH.EQ.1)CALL PPRINT(XJA)

C RESET OFFSET, SCALE AND INTEGRATION STEP SIZE

                CALL UPLOW(ZMN,ZSIG,BALOWER,BUPPER)
                CALL GETSCL(BALOWER,BUPPER,BSCL,BOFF)
                CALL RDOUT(ALOWER,UPPER,BSCL,BOFF,XJA,XJB)
                CALL SET1(ALOWER,BALOWER,1,NDIM)
                CALL SET1(UPPER,BUPPER,1,NDIM)
                CALL SET1(SCALE,BSCL,1,NDIM)
                CALL SET1(OFF,BOFF,1,NDIM)
                CALL SWITCH(XJA,XJB)
                CALL STEPSZ(NSTEP,ALOWER,UPPER,STEP)
 1ØØ            CONTINUE

        WRITE(6,1645)
 8888   WRITE(2Ø,1645)

 1645   FORMAT(1X,' ----------------------------------------')
        CLOSE(UNIT=2Ø)
        STOP
        END

C-----------------------------------------------------------------------

C SUBROUTINE NUMSTEP COMPUTES THE NUMBER OF STEPS

        SUBROUTINE NUMSTEP(XLOW,XUP,XSTEP,NDIM,NSTEP)
        DIMENSION XLOW(1),XUP(1),XSTEP(1),NSTEP(1)

        DO 1 I=1,NDIM
        NSTEP(I)=INT(((XUP(I)-XLOW(I))/XSTEP(I))+.5)
 1      CONTINUE

        RETURN
        END

C-----------------------------------------------------------------------

C SUBROUTINE STEPSZ COMPUTES THE SIZE OF STEPS

        SUBROUTINE STEPSZ(NSTEP,ALOWER,UPPER,NDIM,STEP)
        DIMENSION NSTEP(1),STEP(1),ALOWER(1),UPPER(1)
```

```
      DO 1 I=1,NDIM
      STEP(I)=(UPPER(I)-ALOWER(I))/FLOAT(NSTEP(I))
 1    CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------

C UPLOW COMPUTES BOUNDARY OF PROBABILITY DENSITY

      SUBROUTINE UPLOW(ZMN,ZSIG,BLOW,BUP)
      DIMENSION ZMN(7),ZSIG(7),BLOW(7),BUP(7)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM

      DO 1 I=1,NDIM
      BLOW(I)=-4.*ZSIG(I)+ZMN(I)
      BUP(I)= 4.*ZSIG(I)+ZMN(I)
 1    CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------

C RDOUT RESAMPLES PROBABILITY DENSITY

      SUBROUTINE RDOUT(ALOWER,UPPER,BSCL,BOFF,XJA,XJB)
      DIMENSION ALOWER(7),UPPER(7),BSCL(7),BOFF(7),U(7)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      REAL*8 XJA(50000),VAL XJINT
      REAL*8 XJB(KLEN(1),KLEN(2),KLEN(3),KLEN(4),KLEN(5),
     .           KLEN(6),KLEN(7))

      DO 100 I1=1,KLEN(1)
      DO 100 I2=1,KLEN(2)
      DO 100 I3=1,KLEN(3)
      DO 100 I4=1,KLEN(4)
      DO 100 I5=1,KLEN(5)
      DO 100 I6=1,KLEN(6)
      DO 100 I7=1,KLEN(7)
      CALL CONVRT(I1,I2,I3,I4,I5,I6,I7,
     .            BSCL,BOFF,U)
      VAL=XJINT(XJA,U)
      XJB(I1,I2,I3,I4,I5,I6,I7)=VAL
100   CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------

C CONVERT INDICES TO STATE VARIABLES

      SUBROUTINE CONVRT(I1,I2,I3,I4,I5,I6,I7,
     .                  BSCL,BOFF,U)
      DIMENSION BOFF(7),BSCL(7),U(7)
      U(1)=BOFF(1)+(I1-1)/BSCL(1)
      U(2)=BOFF(2)+(I2-1)/BSCL(2)
      U(3)=BOFF(3)+(I3-1)/BSCL(3)
      U(4)=BOFF(4)+(I4-1)/BSCL(4)
      U(5)=BOFF(5)+(I5-1)/BSCL(5)
      U(6)=BOFF(6)+(I6-1)/BSCL(6)
```

```
        U(7)=BOFF(7)+(I7-1)/BSCL(7)
        RETURN
        END
C-------------------------------------------------------------------------
C UPDATE FILTER AND OUTPUTS
        SUBROUTINE FILTER(XX,QM1,RM1)
        COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
        DIMENSION QM1(7),RM1(10000)
        DIMENSION XX(7),ZZ(10000),YY(7)
        COMMON/DATA/ZT(10000)
C GET NEW STATE
        CALL F(XX,YY,NDIM)
        CALL NOIV(ZZ,QM1,NDIM)
        CALL AVV(YY,ZZ,XX,NDIM)
C GET NEW MEASUREMENTS
        CALL H(XX,ZT,NOUTS)
        CALL NOIV(ZZ,RM1,NOUTS)
        CALL AVV(ZT,ZZ,ZT,NOUTS)
        RETURN
        END
C---------------------------------------------------------------
C COMPUTE NX1 NOISE VECTOR A WITH VAR B
        SUBROUTINE NOIV(A,B,N)
        DIMENSION A(N),B(N)
        DO 100 INDX=1,N
        A(INDX)=GAUSS(0.,SQRT(1./B(INDX)))
 100    CONTINUE
        RETURN
        END
C---------------------------------------------------------------
C ADD VECTOR V1 TO V2 AND STORE IN V3
        SUBROUTINE AVV(V1,V2,V3,N)
        DIMENSION V1(N),V2(N),V3(N)
        DO 100 INDX=1,N
        V3(INDX)=V1(INDX)+V2(INDX)
 100    CONTINUE
        RETURN
        END
C---------------------------------------------------------------
C RESET UPPER AND LOWER BOUNDS
        SUBROUTINE RESET(ALOWER,ZMN,ZSIG,SCL)
        DIMENSION ALOWER(7),ZMN(7),ZSIG(7)
        COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
```

```
      DO 100 INDX=1,NDIM
      ALOWER(INDX)=ZMN(INDX)-SCL*ZSIG(INDX)
100   CONTINUE

      RETURN
      END

C-----------------------------------------------------------------

C GET SCALE AND OFFSET

      SUBROUTINE GETSCL(ALOWER,UPPER,SCALE,OFF)
      DIMENSION ALOWER(7),UPPER(7),SCALE(7),OFF(7)
      COMMON/JDESCR/KLEN(7),XSCALE(7),XOFF(7),NDIM

      DO 100 INDX=1,NDIM
      SCALE(INDX)=FLOAT(KLEN(INDX)-2)/(UPPER(INDX)-ALOWER(INDX))
      OFF(INDX)=ALOWER(INDX)
100   CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------

C INITIALIZE PRIOR PROBABILITY DENSITY

      SUBROUTINE SETUP(XJA,VAL,ILW,IUP,ALOWER,UPPER,ANS)
      DIMENSION IUP(7),ILW(7)
      CHARACTER*5 ANS
      DIMENSION ALOWER(7),UPPER(7)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      REAL*8 XJA(KLEN(1),KLEN(2),KLEN(3),KLEN(4),KLEN(5),
     .                KLEN(6),KLEN(7))

      DO 100 I1=1,KLEN(1)
      DO 100 I2=1,KLEN(2)
      DO 100 I3=1,KLEN(3)
      DO 100 I4=1,KLEN(4)
      DO 100 I5=1,KLEN(5)
      DO 100 I6=1,KLEN(6)
      DO 100 I7=1,KLEN(7)
      XJA(I1,I2,I3,I4,I5,I6,I7)=0.
100   CONTINUE

      IF(ANS.EQ.'YES  '.OR.ANS.EQ.'Y    ')GOTO 2001

C IF UNIFORM

      DO 200 I1=ILW(1),IUP(1)
      DO 200 I2=ILW(2),IUP(2)
      DO 200 I3=ILW(3),IUP(3)
      DO 200 I4=ILW(4),IUP(4)
      DO 200 I5=ILW(5),IUP(5)
      DO 200 I6=ILW(6),IUP(6)
      DO 200 I7=ILW(7),IUP(7)
      XJA(I1,I2,I3,I4,I5,I6,I7)=VAL
200   CONTINUE

      GOTO 1000

C IF GAUSSIAN

2001  DO 201 I1=1,KLEN(1)
```

```
      DO 201 I2=1,KLEN(2)
      DO 201 I3=1,KLEN(3)
      DO 201 I4=1,KLEN(4)
      DO 201 I5=1,KLEN(5)
      DO 201 I6=1,KLEN(6)
      DO 201 I7=1,KLEN(7)
      XJA(I1,I2,I3,I4,I5,I6,I7)=GDENSITY(I1,I2,I3,I4,I5,I6,I7,ILW,IUP,
                                         ALOWER,UPPER)
 201  CONTINUE

 1000 CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------

C COMPUTE GAUSSIAN PROBABILITY DENSITY

      REAL*8 FUNCTION GDENSITY(I1,I2,I3,I4,I5,I6,I7,ILW,IUP,ALOWER,UPPER)
      DIMENSION ALOWER(7),UPPER(7)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      DIMENSION INDEX(7),ILW(7),IUP(7)
      REAL*8 D,ZDEN

      INDEX(1)=I1
      INDEX(2)=I2
      INDEX(3)=I3
      INDEX(4)=I4
      INDEX(5)=I5
      INDEX(6)=I6
      INDEX(7)=I7

      D=1.

      DO 10 I=1,NDIM
      XMU=FLOAT(ILW(I))+FLOAT(IUP(I)-ILW(I))/2.
      SIGMA=FLOAT(IUP(I)-ILW(I))/8.
      X=FLOAT(INDEX(I))
      D=D*ZDEN(X,XMU,SIGMA)*((KLEN(I)-2)/(UPPER(I)-ALOWER(I)))
 10   CONTINUE

      GDENSITY=D
      RETURN
      END

C-----------------------------------------------------------------------

C COMPUTE GAUSSIAN PROBABILITY

      REAL*8 FUNCTION ZDEN(X,XMU,SIGMA)
      REAL*8 D
      D=(X-XMU)/SIGMA
      ZDEN=DEXP(-.5*D*D)/(SIGMA*SQRT(2.*3.14))
      RETURN
      END

C-----------------------------------------------------------------------

C PRINT RESULTS

      SUBROUTINE PPRINT(XJA)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      REAL*8 XJA(KLEN(1),KLEN(2),KLEN(3),KLEN(4),
```

```
     .                    KLEN(5),KLEN(6),KLEN(7)) ◄
          WRITE(6,1000)(((((((XJA(I1,I2,I3,I4,I5,I6,I7),I1=1,KLEN(1)),
     .                                                  I2=1,KLEN(2)),
     .                                                  I3=1,KLEN(3)),
     .                                                  I4=1,KLEN(4)),
     .                                                  I5=1,KLEN(5)),
     .                                                  I6=1,KLEN(6)),
     .                                                  I7=1,KLEN(7))

          WRITE(20,1000)(((((((XJA(I1,I2,I3,I4,I5,I6,I7),I1=1,KLEN(1)),
     .                                                  I2=1,KLEN(2)),
     .                                                  I3=1,KLEN(3)),
     .                                                  I4=1,KLEN(4)),
     .                                                  I5=1,KLEN(5)),
     .                                                  I6=1,KLEN(6)),
     .                                                  I7=1,KLEN(7))

 1000     FORMAT(1X,5D12.5)
          RETURN
          END

C-------------------------------------------------------------------

C SWAP ARRAYS

          SUBROUTINE SWITCH(XJA,XJB)
          COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
          REAL*8 TEMP
          REAL*8 XJA(KLEN(1),KLEN(2),KLEN(3),KLEN(4),
     .                    KLEN(5),KLEN(6),KLEN(7))
          REAL*8 XJB(KLEN(1),KLEN(2),KLEN(3),KLEN(4),
     .                    KLEN(5),KLEN(6),KLEN(7))

          DO 100 I1=1,KLEN(1)
          DO 100 I2=1,KLEN(2)
          DO 100 I3=1,KLEN(3)
          DO 100 I4=1,KLEN(4)
          DO 100 I5=1,KLEN(5)
          DO 100 I6=1,KLEN(6)
          DO 100 I7=1,KLEN(7)
          TEMP=XJA(I1,I2,I3,I4,I5,I6,I7)
          XJA(I1,I2,I3,I4,I5,I6,I7)=XJB(I1,I2,I3,I4,I5,I6,I7)
          XJB(I1,I2,I3,I4,I5,I6,I7)=TEMP
 100      CONTINUE

          RETURN
          END

C-------------------------------------------------------------------

C COMPUTE IMAGE

          SUBROUTINE H(V,HOUT,NOUT)
          DIMENSION V(7),HOUT(10000)
          COMMON/SPECS/NFOVX,NFOVY,DETX,DETY
          COMMON/TARGET/XSIZE,YSIZE,DELTAT,XTAR
          NOUT=NFOVX*NFOVY
          ICNT=0

C TARGET LOCATION

          XTAR=V(1)
          YTAR=-YSIZE/2.+NFOVY*DETY/2.
```

```
      DO 1ØØ J=1,NFOVY
      DO 1ØØ I=1,NFOVX
      ICNT=ICNT+1

C CONVERT PIXELS I,J TO IMAGE UNITS

      YJ=FLOAT(J-1)*DETY-FLOAT(NFOVY-1)*DETY/2.
      XI=FLOAT(I-1)*DETX-FLOAT(NFOVX-1)*DETX/2.

C PIXEL BLUR FACTOR

      XFACT=DETOUT(XTAR,XI,DETX,XSIZE)

C PIXEL OUTPUT

      VAL=XFACT*DELTAT
      HOUT(ICNT)=VAL
 1ØØ  CONTINUE

      RETURN
      END

C-------------------------------------------------------------------------

C COMPUTE PIXEL BLUR FACTOR

      FUNCTION DETOUT(X,PICLOC,BLUR,SIZE)

C TARGET LOCATION IS ZERO

      XX=PICLOC-X

C CHECK TO SEE IF TARGET COVERS ANY OF THIS PIXEL

      ABSXX=ABS(XX)
      IF(ABSXX.GT..5*(SIZE+BLUR))GOTO 5Ø

C IS PIXEL ON LEFT EDGE OF TARGET?

      XLEFT=XX+(SIZE-BLUR)/2.
      IF(XLEFT.GT.Ø.)GOTO 6Ø

C PIXEL IS ON LEFT EDGE OF TARGET SO US TRI. FUNCTION
C TO COMPUTE PIXEL OUTPUT

      ARG=(XLEFT)/BLUR
      DETOUT=TRI(ARG)
      RETURN

C PIXEL NOT ON LEFT EDGE SO CHECK RIGHT EDGE

 6Ø   XRIGHT=XX-(SIZE-BLUR)/2.
      IF(XRIGHT.LT.Ø.)GOTO 7Ø

C PIXEL IS ON RIGHT EDGE OF TARGET SO US TRI. FUNCTION
C TO COMPUTE PIXEL OUTPUT

      ARG=(XRIGHT)/BLUR
      DETOUT=TRI(ARG)
      RETURN

C PIXEL IN MIDDLE REGION OF TARGET

 7Ø   DETOUT=1.
```

```
        RETURN

C PIXEL NOT ON TARGET

 5Ø     DETOUT=Ø.
        RETURN
        END

C-------------------------------------------------------------------------

C TRIANGLE FUNCTION

        FUNCTION TRI(X)
        Y=ABS(X)
        IF(Y.GT.1.)GOTO 1
        TRI=1.-Y
        RETURN
 1      TRI=Ø.
        RETURN
        END

C-------------------------------------------------------------------------

C STATE TRANSITION FUNCTION

        SUBROUTINE F(V,FOUT,NSTATE)
        DIMENSION V(1),FOUT(1)
        NSTATE=1
        FOUT(1)=V(1)
        RETURN
        END

C-------------------------------------------------------------------------

C       MVV
C THIS ROUTINE WILL COMPUTE THE SCALAR TRANPOSE(A)*B*C
C WHERE B IS DIAGONAL.  A,C ARE VECTORS OF DIMENSION N

        SUBROUTINE MVV(A,B,C,N,OUT)
        DIMENSION A(N),B(N),C(N)
        REAL*8 OUT
        OUT=Ø.

C B CONTAINS ONLY THE DIAGONAL ELEMENTS

        DO 1ØØ INDX=1,N
        OUT=OUT+A(INDX)*B(INDX)*C(INDX)
 1ØØ    CONTINUE

        RETURN
        END

C-------------------------------------------------------------

C       DVV
C THIS ROUTINE WILL COMPUTE A-B WHERE
C A,B ARE N DIMENSIONAL VECTORS

        SUBROUTINE DVV(A,B,C,N)
        DIMENSION A(N),B(N),C(N)

        DO 1ØØ INDX=1,N
        C(INDX)=A(INDX)-B(INDX)
 1ØØ    CONTINUE
```

```
      RETURN
      END
C-----------------------------------------------------------------
C       XJINT
C THIS ROUTINE WILL INTERPOLATE THE XJ
C PROBABILITY DENSITY FUNCTION TO GET
C TRUE VALUE. INPUT IS A VECTOR U. OUTPUT
C IS THE SCALAR VALUE.
      REAL*8 FUNCTION XJINT(XJ,U)
      REAL*8 XJ,A,B,XJGRAB
      DIMENSION XJ(5000),ISUBSC(7),DX(7),U(7)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
C GET INDICES
      DO 100 INDX=1,7
C HANDLE HIGHER DIMENSION NOT USED
      IC=1
      IF(INDX.GT.NDIM)GOTO 50
C CTS OF U FOR INDEX
      PLACE=SCALE(INDX)*(U(INDX)-OFF(INDX))
      IC=IFIX(PLACE)+1
C ERROR CHECKING
      IF((IC-1.LT.1).OR.(IC+1.GT.KLEN(INDX)))GOTO 500
C GET DELTA FOR INTERPOLATION
      DX(INDX)=(PLACE-FLOAT(IC-1))
 50   ISUBSC(INDX)=IC
100   CONTINUE
C NOW DO INTERPOLATION
      A=XJGRAB(XJ,ISUBSC)
      DO 200 INDX1=1,NDIM
C INCREMENT LOOKUP INDEX
      ISUBSC(INDX1)=ISUBSC(INDX1)+1
      B=XJGRAB(XJ,ISUBSC)
C INTERPOLATE BY ADDING IN DERIVATIVE
      A=A+(B-A)*DX(INDX1)
C RESTORE SUBSCRIPT
      ISUBSC(INDX1)=ISUBSC(INDX1)-1
200   CONTINUE
C OUTPUT RESULT
      IF(A.LT.0.)A=0.
```

```
      XJINT=A
      RETURN

C ERROR CHECK

 500  CONTINUE

      XJINT=0.
      RETURN
      END

C-----------------------------------------------------------

C     XJGRAB
C FUNCTION TO GET AN ELEMENT FROM THE
C XJ ARRAY, ACCESSED BY THE COEFFICIENTS
C IN THE ISUBSC ARRAY

      REAL*8 FUNCTION XJGRAB(XJ,ISUBSC)
      DIMENSION ISUBSC(7)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      REAL*8 XJ(KLEN(1),KLEN(2),KLEN(3),KLEN(4),
     .          KLEN(5),KLEN(6),KLEN(7))
      XJGRAB=XJ(ISUBSC(1),ISUBSC(2),ISUBSC(3),ISUBSC(4),
     .          ISUBSC(5),ISUBSC(6),ISUBSC(7))
      RETURN
      END

C-----------------------------------------------------------------

C     XJPUT
C SUBROUTINE TO PUT AN ELEMENT IN XJ ARRAY

      SUBROUTINE XJPUT(XJ,VAL,ISUBSC)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      DIMENSION ISUBSC(7)
      REAL*8 VAL
      REAL*8 XJ(KLEN(1),KLEN(2),KLEN(3),KLEN(4),
     .          KLEN(5),KLEN(6),KLEN(7))
      XJ(ISUBSC(1),ISUBSC(2),ISUBSC(3),ISUBSC(4),ISUBSC(5),
     .   ISUBSC(6),ISUBSC(7))=VAL
      RETURN
      END

C-------------------------------------------------------------------

C     UPDATE
C UPDATE PASSES THROUGH AN ITERATION OF
C THE FILTER EQUATION
C IS SETS UP XJB FROM XJA AND PRODUCES MEANS AND SIGMAS

      SUBROUTINE  UPDATE(ALOWER,UPPER,STEP
     .                   ,ZMN,ZSIG)
      REAL*8 XJA,XJB,VAL,ZNUM,SUMINT,FUNUM,FX1,FX2
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
      COMMON/VARS/U(7),XJA(50000),XJB(50000)
      COMMON/IVARA/IVAR
      DIMENSION JSUBSC(7),IONE(7)
      DIMENSION ALOWER(7),UPPER(7),STEP(7),ZMN(7)
      DIMENSION ZSIG(7)
      DATA IONE/7*1/
      EXTERNAL FUNUM,FX1,FX2

C INITIALIZE JSUBSC,AND U
```

```
      CALL SET1(JSUBSC,IONE,1,7)
      CALL SET1(U,ALOWER,1,7)
100   LDIM=1

C CALCULATE NUMERATOR OF FILTER EXPRESSION

      ZNUM=SUMINT(FUNUM,NDIM,ALOWER,UPPER,STEP)
      CALL XJPUT(XJB,ZNUM,JSUBSC)

C INCREMENT SUBSCRIPT AND U

200   JSUBSC(LDIM)=JSUBSC(LDIM)+1
      U(LDIM)=U(LDIM)+1./SCALE(LDIM)

C FINISH THIS DIMENSION?

      IF(JSUBSC(LDIM).LE.KLEN(LDIM))GOTO 100

C YES-INCREMENT DIMENSION-RESET U

      JSUBSC(LDIM)=1
      U(LDIM)=ALOWER(LDIM)
      LDIM=LDIM+1

C ARE YOU FINISHED WITH ALL DIMENSIONS?

      IF(LDIM.LE.NDIM)GOTO 200

C NORMALIZE PROBABILITY DENSITY

      CALL NORMLZE(ALOWER,UPPER,STEP,XJB)

C  YES-GET MEAN AND SIGMA
C TAKE MEANS AND SIGMAS OF STATE VARIABLES

      DO 300 INDX=1,NDIM

C SET IVAR TO THE BVARIABLE INDEX TO BE OPERATED ON

      IVAR=INDX

C FX1 CONSISTS OF U*J(U)

      ZVR=SUMINT(FX2,NDIM,ALOWER,UPPER,STEP)
      ZMN(INDX)=SUMINT(FX1,NDIM,ALOWER,
     .                 UPPER,STEP)

C CALCULATE SIGMA

      IF(ZVR-ZMN(INDX)*ZMN(INDX).LT.0.)WRITE(6,1212)
1212  FORMAT(1X,' NEG SIGMA SHIT ')
      ZSIG(INDX)=SQRT((ABS(ZVR-ZMN(INDX)*ZMN(INDX))))
300   CONTINUE

      RETURN
      END

C-------------------------------------------------------------

C NORMALIZE PROBABILITY DENSITY

      SUBROUTINE NORMLZE(ALOWER,UPPER,STEP,XJB)
      COMMON/JDESCR/KLEN(7),SCALE(7),OFF(7),NDIM
```

```
      REAL*8 XJB(KLEN(1),KLEN(2),KLEN(3),KLEN(4),
     .                KLEN(5),KLEN(6),KLEN(7))
      DIMENSION UPPER(7),ALOWER(7),STEP(7)
      REAL*8 DENSITY,SUM,HTERM
      EXTERNAL DENSITY

      DO 2000 I1=1,KLEN(1)
      DO 2000 I2=1,KLEN(2)
      DO 2000 I3=1,KLEN(3)
      DO 2000 I4=1,KLEN(4)
      DO 2000 I5=1,KLEN(5)
      DO 2000 I6=1,KLEN(6)
      DO 2000 I7=1,KLEN(7)
      XJB(I1,I2,I3,I4,I5,I6,I7)=XJB(I1,I2,I3,I4,I5,I6,I7)
     .                *HTERM(I1,I2,I3,I4,I5,I6,I7,NDIM,SCALE,OFF)
2000  CONTINUE

      SUM=SUMINT(DENSITY,NDIM,ALOWER,UPPER,STEP)

      DO 200 I1=1,KLEN(1)
      DO 200 I2=1,KLEN(2)
      DO 200 I3=1,KLEN(3)
      DO 200 I4=1,KLEN(4)
      DO 200 I5=1,KLEN(5)
      DO 200 I6=1,KLEN(6)
      DO 200 I7=1,KLEN(7)
      XJB(I1,I2,I3,I4,I5,I6,I7)=XJB(I1,I2,I3,I4,I5,I6,I7)/SUM
200   CONTINUE

      RETURN
      END

C-------------------------------------------------------------

C CURRENT CONDITIONAL PROBABILITY DENSITY

      REAL*8 FUNCTION DENSITY(X)
      DIMENSION X(7)
      REAL*8 XJA,XJB,XJINT
      COMMON/VARS/U(7),XJA(50000),XJB(50000)
      DENSITY=XJINT(XJB,X)
      RETURN
      END

C-------------------------------------------------------------

C PERFORM NUMERICAL INTEGRATION IN UP TO 10 DIMENSIONS
C OF ARBITRARY FUNCTION .
C INPUTS:
C       FUNC    FUNCTION TO BE INTEGRATED. MUST BE DEFINED AS
C               AN EXTERNAL IN USERS CALLING ROUTINE.
C       NDIM    DIMENSIONS OF SPACE TO BE INTERGATED OVER.
C       FROM    NDIM DIMENIONAL VECTOR WHICH SPECIFIES THE LOWER
C               INTERGATION LIMIT FOR EACH DIMENSION.
C       TO      NDIM DIMENSIONAL VECTOR WHICH SPECIFIES THE UPPER
C               INTEGRATION LIMIT FOR EACH DIMENSION.
C       STEP    NDIM DIMENISIONAL VECTOR WHICH SPECIFICES
C               THE INTERGATION STEP SIZE FOR EACH DIMENSION.

      REAL*8 FUNCTION SUMINT(FUNC,NDIM,FROM,TO,STEP)
      REAL*8 FUNC,TOTAL,ZERO
      DIMENSION FROM(10),TO(10),STEP(10)
      DIMENSION X(10),ZERO(0:10)
      DIMENSION TOTAL(0:10)
```

```
        DATA ZERO/11*0./
C INITIAL TOTAL TO ZERO
        CALL SET(TOTAL,ZERO,0,NDIM)
C INITIALIZE X- VARIBLE OF INTEGRATION.
        CALL SET1(X,FROM,1,NDIM)
 100     IDIM=1
C COMPUTE INTERGRAN
        TOTAL(0)=FUNC(X)
C INCREMENT X
 200    X(IDIM)=X(IDIM)+STEP(IDIM)
        TOTAL(IDIM)=TOTAL(IDIM)+TOTAL(IDIM-1)*STEP(IDIM)
C RESET LOWER DIMENSION SUM
        TOTAL(IDIM-1)=0.
        IF(X(IDIM).LT.TO(IDIM))GOTO 100
C REINITILIZE LOWER DIMENSION
        X(IDIM)=FROM(IDIM)
C GO ONE DIMENSION HIGHER
        IDIM=IDIM+1
C FINSHED?
        IF(IDIM.LE.NDIM)GOTO 200
C GET SUM
        SUMINT=TOTAL(NDIM)
        RETURN
        END
C-----------------------------------------------------------------------
C SET VECTOR X TO VECTOR Y, DOUBLE PRECISION
        SUBROUTINE SET(X,Y,NSTART,NSTOP)
        REAL*8 X(NSTART:NSTOP),Y(NSTART:NSTOP)
        DO 100 I=NSTART,NSTOP
        X(I)=Y(I)
 100    CONTINUE
        RETURN
        END
C-----------------------------------------------------------------------
C SET VECTOR X TO VECTOR Y, SINGLE PRECISION
        SUBROUTINE SET1(X,Y,NSTART,NSTOP)
```

```
      DIMENSION X(NSTART:NSTOP),Y(NSTART:NSTOP)

      DO 100 I=NSTART,NSTOP
      X(I)=Y(I)
100   CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------

C FUNUM-INTEGRAND FUNCTION FOR FILTER'S NUMERATOR

      REAL*8 FUNCTION FUNUM(V)
      REAL*8 XJA,XJB,T4,T5,DD,DT1,DT2,XJINT,T2
      DIMENSION V(7),HOUT(100000),FOUT(7),TEMP(100000)
      COMMON/VARS/U(7),XJA(500000),XJB(500000)
      COMMON/DATA/ZT(100000)
      COMMON/NOISE/QM1(7),RM1(100000)

C COMPUTE SECOND TERM

      CALL F(V,FOUT,NSTATE)
      CALL DVV(U,FOUT,UMV,NSTATE)
      CALL MVV(UMV,QM1,UMV,NSTATE,T2)

      DT2=T2
      DD=DEXP(-.5*(DT2))

C LOOK UP OLD PROBABILITY

      T4=XJINT(XJA,V)

      T5=T4*DD
      FUNUM=T5

      RETURN
      END

C-----------------------------------------------------------------

      REAL*8 FUNCTION HTERM(I1,I2,I3,I4,I5,I6,I7,NDIM,SCALE,OFF)
      REAL*8 DT1
      COMMON/DATA/ZT(100000)
      COMMON/NOISE/QM1(7),RM1(100000)
      DIMENSION SCALE(7),OFF(7),U(7)
      DIMENSION TEMP(100000),HOUT(100000)
      CALL CONVRT(I1,I2,I3,I4,I5,I6,I7,SCALE,OFF,U)
      CALL H(U,HOUT,NOUTS)
      CALL DVV(ZT,HOUT,TEMP,NOUTS)
      CALL MVV(TEMP,RM1,TEMP,NOUTS,T1)
      DT1=T1
      HTERM=DEXP(-.5*DT1)
      RETURN
      END

C-----------------------------------------------------------------

      REAL*8 FUNCTION FX1(V)
      DIMENSION V(7)
      REAL*8 XJA,XJB,XJINT
      COMMON/VARS/U(7),XJA(500000),XJB(500000)
      COMMON/IVARA/IVAR
      FX1=XJINT(XJB,V)*V(IVAR)
```

```
        RETURN
        END
C-------------------------------------------------------------
        REAL*8 FUNCTION FX2(V)
        DIMENSION V(7)
        REAL*8 XJA,XJB,XJINT
        COMMON/VARS/U(7),XJA(5000),XJB(5000)
        COMMON/IVARA/IVAR
        FX2=XJINT(XJB,V)*V(IVAR)*V(IVAR)
        RETURN
        END
C-------------------------------------------------------------
C THIS FUNCTION USE THE REJECTION-ACCEPTANCE
C METHOD TO GENERATE A GAUSSIAN(XMEAN,SIGMA)
C RANDOM VARIBLE.
C
        FUNCTION GAUSS(XMEAN,SIGMA)
        COMMON/SEED/ISEED
C
C GENERATE TWO UNIFORM(0,1)
C
 1      U1=RAN(ISEED)
        U2=RAN(ISEED)
        X=-ALOG(U1)
        IF(U2.GT.EXP(-(X-1.)**2/2.))GOTO 1

C
C X HAS BEEN ACCEPTED. GENERATE
C RANDOM SIGN AND ATTACH TO X.
        P=RAN(I)
        SIGN=1.
        IF(P.GT..5)SIGN=-1.
        X=SIGN*X
C
C X IS NOW GAUSS(0,1). CONVERT TO
C GAUSS(XMEAN,SIGMA).
        GAUSS=SIGMA*X+XMEAN
        RETURN
        END
C-------------------------------------------------------------
        SUBROUTINE SET_SEED(I)
        CHARACTER*8 TM
        COMMON/SEED/ISEED
        DATA ISEED/99999/
C IF I<= 0 SET SEED FROM CLOCK
C OTHERWISE SEED IS SET TO USUER SUPPLIED VALUE I
        IF(I.GT.0)GOTO 10
        CALL TIME(TM)
        I1=ICHAR(TM(8:8))
        ISEED=2*(I1*I1+100)+1
        RETURN
C
 10     ISEED=I
        RETURN
        END
```

REFERENCES

1. Kushner, H.J., Stochastic Stability and Control, 1967, Academic Press

2. McGarty, T.P., Stochastic Systems and State Estimation, 1974, John Wiley

3. Schuss, A., Theory and Applications of Stochastic Differential Equations, 1980, John Wiley

4. Jazwinski, A.H., Stochastic Processes and Filtering Theory, 1970, Academic Press

5. Astrom, K.J., Introduction to Stochastic Control Theory, 1970, Academic Press

6. Stratonovich, R.L., Conditional Markov Processes and Their Application to the Theory of Optimal Control, 1968, American Elsevier Pub. Co.

7. Srinivansan, S.K. and Vasudevan, R., Introduction to Random Differential Equations and Their Applications, 1971, American Elsevier Pub. Co.

8. Maybeck, P.S., Stochastic Models, Estimation, and Control, Vol. 1, 1971, Academic Press

9. Parzen, E., Stochastic Processes, 1962, Holden-Day Inc.

10. Doob, J.L., Stochastic Process, 1953, John Wiley

11. Ito, K. and McKena, H.P., Diffusion Processes and Their Sample Paths, 1965, Academic Press

12. Bellman, R., Methods of Nonlinear Analysis, Vol. I, 1970, and Vol. II, 1973, Academic Press

13. Fisher, J.R., Optimal Nonlinear Filtering; in Advances in Control Systems, Vol. 5, C.T. Leondes, Edition, 1967, Academic Press

14. Bejczy, A.K. and Sridhar, R., Analytic Methods for Performance Evaluation of Nonlinear Filters, Symposium on Nonlinear Estimation and Its Applications, IEEE, #70C66-AC, 1970

15. Mercier, D.E., An Extended Kalman Filter for Use in a Shared Aperture Medium Range Tracker, 1978 Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio

16. Jensen, R.L. and Harnty, D.A., An Adaptive Distributed-Measurement Extended Kalman Filter for a Short Range Tracker (Vol. 1 and 2), 1979 Thesis, Air Force Institute

17. Gilman, A.S. and Rhodes, I.B., "Cone-Bounded Nonlinearities and Mean-Square Bounds — Estimation Upper Bound," IEEE Trans. Automat. Contr., Vol. AC-18, pp. 260-265, June 1973

18. Dunn, K.P. and Rhodes, I.B., "Cone-Bounded Nonlinearities and Mean-Square Bounds — Smoothing and Prediction," Automatica, Sept. 1975; also in Proc. Int. Fed. Automat. Contr., Congr., Boston-Cambridge, Mass., Aug. 1975

19. Gilman, A.S. and Rhodes, I.B., "Cone-Bounded Nonlinearities and Mean-Square Bounds — Quadratic Regulation Bounds," Dep. Syst. Sci., Math., Washington Univ., St. Louis, MO, Rep. SSM 7503, June 1975

20. Snyder, D.L. and Rhodes, I.B., "Filtering and Control Performance Bounds with Implications on Asymptotic Separation," Automatica, Vol. 8, pp. 747-753, Nov. 1972

21. Snyder, D.L. and Rhodes, I.B., "Phase and Frequency Tracking Accuracy of Direct-Detection Optical Communication Systems," IEEE Trans. Commun., Vol. COM-20, pp. 1139-1142, Dec. 1972.

22. Gilman, A.S. and Rhodes, I.B., "Cone-Bounded Nonlinearities and Mean-Square Bounds — Estimation Lower Bound," IEEE Trans. on Automatic Control, Vol. AC-20, No. 5, Oct. 1975

23. Bobrovsky, B.Z. and Zakai, Moshe, "A Lower Bound on the Estimation Error for Markov Processes," IEEE Trans. Automat. Contr., pp. 786-788, December 1975

24. Galdos, J.I., "A Cramer-Rao Bound for Multidimensional Discrete-Time Dynamical Systems," IEEE Trans. Automat. Contr., Vol. AC-25, No. 1, pp. 117-119, February 1980

25. Zakai, M., "Lower and Upper Bounds on the Optimal Filtering Error of Certain Diffusion Processes, " IEEE Trans. on Information Theory, Vol. IT-18, No. 3, pp. 325-331, May 1972

26. Hibey, J.L., "Performance Bounds for Differential Systems Incurring Abrupt Random Changes," IEEE Trans. on Automat. Contr., Vol. AC-26, No. 2, pp. 553-554, April 1981

27. Reischer, CPT. Benjamin, "Assessment of Target Tracking Techniques," SPIE Vol. 178 Smart Sensors (1979)

28. Willett, T.J. and Raimondi, P.K., "Intelligent Tracking Techniques — A Progress Report," SPIE Vol. 178 Smart Sensors (1979)

29. Hall, J.T., "Advanced Target Acquisition and Tracking Concepts for Real-Time Applications," Preprint

30. Frey, R.L., et.al., "ATAC Autocuer Modeling Analysis," Final Report on Contract DAAK70-77-C-0232

31. Maybeck, P.S. (Air Force Institute of Technology), Jensen, R.L., and Harnly, D.A., "An Adaptive Extended Kalman Filter for Target Image Tracking," IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-17, No. 2, March 1981, pp. 173-180

32. Maybeck, P.S. and Mercier, D.E., "A Target Tracker Using Spatially Distributed Infrared Measurements," IEEE Trans. Automat. Contr., Vol. AC-25, April 1980, pp. 222-225

33. Maybeck, P.S., Harnly, D.A., and Jensen, R.E., "Robustness of a New Infrared Target Tracker," Proc. 1980, IEEE Nat. Aerosp. Electron. Conf., (Dayton, OH, May 1980)

34. Fawcett, James (Electronics Lab., G.E. Co.), "A Comparison of Image Tracking Techniques Using Correlation," National Aerospace Electronics Conference, IEEE NAECON 1980, pp. 892-899

35. Sullivan, D.R. and Martin, J.S. (MIT Lincoln Lab.), "A Coarse Search Correlation Tracker for Image Registration," IEEE Trans. on Aerospace and Electronic Systems, AES-17, No. 1, 1981, pp. 29-34.

36. Astrom, K.J. and Eykhoff, P., "System Identification — A Survey," Automatica 7, Pergamon Press, 1971, pp. 123-162.

37. Maybeck, P.S., "Combined Estimation of States and Parameters for On-Line Applications," Ph.D. Dissertation, MIT, Cambridge, MA, Rep T-557, (C.S. Draper Lab.), February 1972

38. Yared, K.I., "On Maximum Likelihood Identification of Linear State Space Models," Ph.D. Dissertation, MIT, Cambridge, MA, Rep LIDS-TH-920, July 1979

39. Duda, R.O. and Hart, P.E., "Pattern Classification and Scene Analysis," Stanford Research Institute, Menlo Park, CA, Wiley-Interscience Publication, 1973

40. Gopinath, K., Stochastic Filtering Theory, 1980, Springer-Verlag

41. Meriri, A.Z. and Porat, B., "Nonlinear Estimation of Generalized Vector Slot Processes," IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-17, No. 4, July 1981

42. VanTrees, H.L., Detection, Estimation, Modulation Theory, Part I, 1968, John Wiley and Sons

43. Kushner, H.J., "Dynamical Equations for the Optimal Nonlinear Filter," Journal of Differential Equations 3,179-190, 1967

44. Private Communication with Dr. Sheldon Katz of the University of Utah


ATE
MED